Campos finitos y AES

José Galaviz Casas

Facultad de Ciencias, UNAM

Índice general

1. Introducción

En general en computación y en particular en criptografía, los campos finitos tienen una gran importancia. Esta proviene del hecho de que en una computadora digital cualquiera, la memoria es un recurso finito, así que es necesario trabajar sobre conjuntos de datos que tengan un dominio finito. Además deseamos hacer operaciones con ellos, calcular cosas, así que necesitamos también que al operar con los datos en cierto dominio, los resultados permanezcan en él. Si se están usando 32 bits para representar un cierto subconjunto de los números enteros es necesario que, dados cualesquiera dos elementos en ese dominio, el resultado de sumarlos, restarlos, multiplicarlos o dividirlos (suponiendo un denominador distinto de cero), siga estando en el conjunto de los enteros representables en 32 bits porque no podemos permitir que las operaciones que realicemos pretendan usar una cantidad arbitraria de memoria para almacenar los resultados que se pueden obtener. No sería plausible pensar en hacer programas que almacenen o intercambien datos cuyo tamaño puede, potencialmente, exceder cualquier cantidad de memoria.

Es en este contexto que las estructuras algebraicas con un número finito de elementos cobran relevancia, en particular aquellas en las que se puede llevar a cabo cualquier operación con la certeza de que estará bien definida, de que necesariamente se obtendrá un resultado válido en el esquema de representación que se esté usando, sin excepciones, sin casos especiales (salvo aquellos que involucran al cero ocasionalmente), sin condiciones que hagan que los programas terminen abrupta e indebidamente su ejecución. La estructura algebraica más completa a la que podemos aspirar, en la que todas las operaciones elementales que deseamos se pueden llevar a cabo, es el campo finito.

2. Definiciones

Comenzaremos por definir las estructuras algebraicas que serán utilizadas.

Definición 1. Un grupo es una pareja $\langle G, \circ \rangle$ donde G es un conjunto no vacío de elementos y $\circ$ es una operación que combina dos elementos cualesquiera de G con las siguientes propiedades. Para cualesquiera $a, b, c \in G$:

1. G es cerrado bajo $\circ$, es decir $a \circ b \in G$.
2. La operación es asociativa, es decir $a \circ (b \circ c) = (a \circ b) \circ c$.
3. Existe un elemento distinguido $e \in G$, llamado neutro o identidad, tal que: $a \circ e = e \circ a = a$.
4. Para todo elemento a existe un elemento $a' \in G$ tal que $a \circ a' = a' \circ a = e$
5. Si además de los requisitos anteriores se cumple que $a \circ b = b \circ a$ para cualesquiera $a, b \in G$, entonces se dice que el grupo $\langle G, \circ \rangle$ es abeliano.

A la cardinalidad (número de elementos) en el grupo se le denomina *orden del grupo*.

Típicamente si la operación de un grupo se denota con un "+" y por tanto al neutro se lo denota como 0 y al inverso de a como $-a$, se dice que se trata de un *grupo aditivo*. Si en cambio la operación es denotada como una simple yuxtaposición o con un "$\cdot$" entonces se trata de un *grupo multiplicativo*, la identidad se denota con un 1 y el inverso de a como a^{-1}. Pero hay que señalar que la distinción es puramente formal, un grupo es un grupo sin importar si su operación se llama multiplicación o suma o alguna otra cosa.

Definición 2. Un campo es una terna $\langle F, +, \cdot \rangle$ donde F es un conjunto no vacío de elementos junto con dos operaciones $+$ y $\cdot$ con las siguientes propiedades.

1. Los elementos de F forman un grupo aditivo con la operación suma "+", se tiene un neutro aditivo denotado con 0 y el inverso aditivo de $a \in F$ denotado como $-a$.

2. Los elementos de F forman un grupo multiplicativo con la operación producto "$\cdot$", se tiene un neutro multipĺicativo denotado con 1 y el inverso multiplicativo de $a \in F$ denotado como a^{-1}.

3. El producto distribuye a la suma. Si $a, b, c \in F$ son tres elementos cualesquiera: $a \cdot (b + c) = a \cdot b + a \cdot c$

Por supuesto el *orden del campo* es también el número de elementos en su grupo aditivo o en su grupo multiplicativo más uno (porque se quita el cero).

Estamos acostumbrados a trabajar en un campo, el de los reales. En este se puede estar confiado en que, sin importar la operación que se haga, el resultado nunca se saldrá del conjunto en el que se está trabajando. En efecto, se pueden sumar, restar, multiplicar y dividir cualesquiera dos números reales (salvo que el cero no puede fungir como divisor) y el resultado será siempre otro número real. Hay que señalar que, dividir un elemento a entre otro b significa, esencialmente, multiplicar a por el inverso multiplicativo de b.

Estas propiedades, particularmente la existencia del inverso multitplicativo, no son triviales, sin embargo uno sólo las aprecia hasta que carece de ellas en algún otro conjunto. Se mostrarán aquí algunos ejemplos.

Ejemplo 2.0.1. En los enteros módulo 6, $\mathbb{Z}_6$, sólo el 1 y el 5 tienen inverso y, son ellos mismos respectivamente:

$$1 \times 1 = 1 \equiv 1 \pmod 6$$

y

$$5 \times 5 = 25 \equiv 1 \pmod 6$$

De hecho la tabla de multiplicar en $\mathbb{Z}_6$ es:

	0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

En cambio en $\mathbb{Z}8$ la tabla de multiplicar es:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Como se puede ver en $\mathbb{Z}_8$ tienen inverso el 1, el 3, el 5 y el 7.

$\lhd$

Tomando esto en consideración pensemos en el inverso multiplicativo de un elemento en un conjunto $\mathbb{Z}_m$. Encontrar el inverso de un número $a \in \mathbb{Z}_m$ consiste en hallar $x \in \mathbb{Z}_m$ tal que $ax = 1$ tomando el producto módulo m. Es decir:

$$ax \equiv 1 \pmod{m} \tag{2.1}$$

Esta ecuación tiene solución, y es única, si y sólo si a y m son primos relativos, es decir $\text{MCD}\,(a, m) = 1$.

Lema 1. *Si $r|a$ y $r|b$, entonces $r|(\alpha\, a + \beta\, b)$ para todo $\alpha, \beta \in \mathbb{Z}$.*

Demostración. Existe x tal que $rx = a$, y existe y tal que $ry = b$. Por lo tanto,

$$\alpha\, a + \beta, b = \alpha\, rx + \beta\, ry = (\alpha\, x + \beta\, y)r,$$

de manera que $r|(\alpha\, a + \beta\, b)$, como aseguramos. $\square$

Teorema 1. *Un elemento a de $\mathbb{Z}_m$ posee inverso multiplicativo módulo m si y sólo si* $\text{MCD}\,(a, m) = 1$.

Demostración. Supongamos primero que $b \in \mathbb{Z}$ es tal que $ab \equiv 1 \pmod{m}$. Entonces $m|(ab - 1)$, es decir, existe una $\beta \in \mathbb{Z}$ tal que $\beta\, m = ab - 1$. Por lo tanto, $ab - \beta\, m = 1$, de manera que el máximo común divisor de a y m debe dividir al 1; la única manera es si $\text{mcd}(a, m) = 1$.

Viceversa, supongamos que $a \in \mathbb{Z}_m$, y que $\text{mcd}(a, m) = 1$. Entonces existen $\alpha, \beta \in \mathbb{Z}$ tales que

$$\alpha\, a + \beta\, m = 1.$$

Entonces $-\beta\, m = \alpha\, a - 1$, es decir, $m | \alpha\, a - 1$. Por lo tanto, $\alpha\, a \equiv 1 \pmod{m}$, de manera que α es un inverso multiplicativo módulo m de a. $\square$

3. Campos primos

Ahora bien ¿Qué se requiere para que en $\mathbb{Z}_m$ *todos* los elementos tuvieran inverso multiplicativo? Por lo que acabamos de ver se necesita que todos fueran primos relativos con el tamaño del módulo m. Pero eso sólo es posible si el módulo es un número primo, porque entonces todos los elementos que le son menores, son primos relativos con él.

Así pues, para que $\mathbb{Z}_m$ sea un campo, necesitamos que m sea primo. Para recordar mejor esto, normalmente denotaremos con $\mathbb{Z}_p$ al campo cuyo módulo es el primo p.

A diferencia de $\mathbb{R}$, un campo $\mathbb{Z}_p$ es finito. En computación esto es particularmente importante dado que en una computadora la memoria es finita, así que podemos tener un conjunto en el que realmente podemos asegurar que las operaciones de suma, resta, multiplicación y división pueden hacerse con la confianza de no exceder el esquema de representación usado (*overflow*). Sólo habría que tener capacidad para representar y operar números enteros del tamaño adecuado luego de elegir a p. En criptografía p suele elegirse muy grande, de miles de dígitos, así que se utilizan bibliotecas especiales para manipular números enteros muy grandes y hacer operaciones modulares.

Ejemplo 3.0.1. Dado que 7 es un número primo, $\mathbb{Z}_7$ es un campo con un número finito de elementos, a saber $\{0, 1, 2, 3, 4, 5, 6\}$. Las tablas de abajo describen las operaciones en el campo.

Suma	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Mult	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

En cada columna de la tabla de multiplicación (salvo la del cero) hay un uno, es decir, siempre hay un elemento tal que al multiplicarlo por quien encabeza la columna resulta 1: todos tienen inverso, salvo el cero, claro.

$\triangleleft$

En campos muy grandes, en los que p es un número de miles de bits de tamaño, generar la tabla de multiplicación no resulta práctico. En ese contexto ¿cómo encontrar el inverso multiplicativo de alguien? El algoritmo más antiguo de la humanidad es la respuesta: el algoritmo de Euclides para encontrar el máximo común divisor de dos números. En el futuro retomaremos el algoritmo, pero por lo pronto recordemos la versión más común de los cursos elementales.

Algoritmo 1 Algoritmo de Euclides para el máximo común divisor.

Require: $r_0 > r_1$

```
function MCD(r_0, r_1)
    if r_1 = 0 then
        return r_0
    else
        return MCD(r_1, r_0 (mód r_1))
    end if
end function
```

Recordemos que en un campo primo, cualquier elemento a del campo es primo relativo con el módulo p, así que $\text{MCD}\,(a, p) = 1$. El algoritmo de Euclides, por tanto, regresará 1 como resultado, dado que es el último residuo del proceso. Pero entonces será posible despejar cada residuo e ir substituyendo el valor de los previos en la expresión resultante para el último, esto nos permitirá escribir el máximo común divisor como una combinación lineal de los números que entraron originalmente al proceso: a y p, algo como:

$$1 \equiv \alpha\ a + \beta\ p \pmod{p}$$

en esta ecuación el segundo sumando de la derecha es múltiplo de p y por tanto es congruente con cero módulo p, de donde:

$$1 \equiv \alpha\ a \pmod{p}$$

el coeficiente del término con a resulta ser, entonces, el inverso multiplícativo de a.

Ejemplo 3.0.2. Aplicaremos el algoritmo de Euclides en $\mathbb{Z}_7$.

Calculemos el $\text{MCD}\,(7,3) = 1$.

$$\begin{aligned} 7 &= 2 \cdot 3 + 1 \\ 3 &= 3 \cdot 1 + 0 \end{aligned} \tag{3.1}$$

uno es el último residuo distinto de cero, así que, despejándolo de 3.1:

$$1 = -2 \cdot 3 + 1 \cdot 7 \tag{3.2}$$

Hay que señalar que $-2 \equiv 5 \pmod{7}$ así que 3.2 es equivalente a:

$$1 \equiv 5 \cdot 3 + 1 \cdot 7 \pmod{7}$$

como el segundo sumando es múltiplo del módulo entonces se anula:

$$1 \equiv 5 \cdot 3 \pmod{7}$$

como podemos corroborar en la tabla anterior de multiplicación en $\mathbb{Z}_7$, el 5 es el inverso multiplicativo del 3.

Si hacemos lo mismo con $\text{MCD}\,(7,4) = 1$.

$$\begin{aligned} 7 &= 1 \cdot 4 + 3 \\ 4 &= 1 \cdot 3 + 1 \\ 3 &= 1 \cdot 3 + 0 \end{aligned}$$

de donde:

$$\begin{aligned} 1 &= 4 - 1 \cdot 3 \\ &= 4 - 1 \cdot (7 - 1 \cdot 4) \\ &= 2 \cdot 4 - 1 \cdot 7 \end{aligned}$$

así que el inverso del 4 es, en efecto, el dos.

$\lhd$

4. Campos de extensión $GF(2^n)$

Los campos primos presentados en la sección previa no son los únicos campos finitos. De hecho hay una infinidad de ellos. Existe exactamente uno (salvo isomorfismo) con p^n elementos, para todo número primo positivo p y n entero positivo, a p en este caso se le denomina la *característica del campo* y de este, se dice que es una extensión del campo con p elementos. Nosotros estamos ahora particularmente interesados en campos de extensión de característica 2, es decir de orden 2^n. Ciertamente estos no los podremos representar como los enteros módulo algún número, porque 2^n no es primo, así que el resultado no sería un campo. Al campo finito con p^n elementos se le denota como $GF(p^n)$ donde GF significa *Galois Field*, en honor a Evariste Galois, a quien debemos el germen de la teoría de los campos finitos.

Para representar a $GF(2^n)$ se pueden usar, sin embargo, polinomios como los elementos del campo y usar, el equivalente en el mundo de los polinomios, de los números primos: los polinomios irreducibles. En este caso los elementos del campo serán polinomios de grado estrictamente menor que n (es decir, a lo más de grado $n-1$) con coeficientes binarios.

Definición 3. Un polinomio irreducible $P(x)$ sobre el campo F es un polinomio no constante con coeficientes en F que no puede ser factorizado como el producto de polinomios de grado menor no constantes con coeficientes en F.

Ejemplo 4.0.1. Usando el campo finito $GF(2)$.

$$(x+1)(x+1) = x^2 + 2x + 1 = x^2 + 1 \pmod 2$$

esta última igualdad podría parecer extraña, pero se basa en el hecho de que estamos trabajando en el campo $GF(2)$, así que las sumas se realizan módulo 2 (y por tanto son equivalentes a las restas), así que el coeficiente 2 del segundo término del polinomio se anula. Por cierto esto significa que el polinomio $x^2 + 1$ no es irreducible, dado que se puede expresar como el producto de $(x + 1)$ por él mismo.

En cambio el polinomio $x^2 + x + 1$ en $GF(2)$ es irreducible, de hecho es el único polinomio irreducible de segundo grado con coeficientes en $GF(2)$. $\lhd$

El campo finito $GF(2^8)$, por ejemplo, tiene exactamente 256 elementos y pueden representarse como polinomios de grado menor o igual a 7:

$$A(x) = a_7x^7 + a_6x^6 + \cdots + a_1x + a_0$$

donde $a_i \in \{0, 1\}$.

Este polinomio se puede representar también como un vector de coeficientes binarios:

$$A = (a_7, a_6, \ldots a_1, a_0)$$

con frecuencia se omitirán las comas en el vector e incluso se escribirá el polinomio usando notación hexadecimal.

4.1. Suma y resta

La suma y resta en el campo finito se llevarán a cabo como las respectivas operaciones entre polinomios considerando que los coeficientes se operan módulo 2.

Sean $A(x), B(x) \in GF(2^n)$ con $A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$ y $B(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1x + b_0$.

- Suma.

  $$S(x) = A(x) + B(x) = \sum_{i=0}^{n-1} c_ix^i$$

  donde $c_i = a_i + b_i \pmod 2$.

- Diferencia.

  $$D(x) = A(x) - B(x) = \sum_{i=0}^{n-1} c_ix^i$$

  donde $c_i = a_i - b_i \pmod 2 = a_i + b_i \pmod 2$. Es decir, la suma y la resta son lo mismo.

Ejemplo 4.1.1. Sumemos dos polinomios en $GF(2^8)$. Sean $A(x) = x^7 + x^6 + x^4$ y $B(x) = x^4 + x^2 + 1$ entonces la suma

$$\begin{array}{rcccc} A(x) = & x^7 + x^6 & + x^4 & & + 1 \\ +B(x) = & & + x^4 & + x^2 & + 1 \\ = C(x) = & x^7 + x^6 & & + x^2 & \end{array}$$

$\triangleleft$

4.2. Producto

Hasta aquí no hay problemas. Estos surgen cuando se considera el producto de dos polinomios, ambos en $GF(2^n)$.

$$P(x) = A(x) \cdot B(x) = \sum_{i=0}^{2n-2} c_i x^i$$

donde

$$c_i = \sum_{j=0}^{i} a_j \, b_{i-j} \pmod 2$$

Aún cuando A y B son de grado, a lo más $n-1$, el producto tendría un grado potencialmente mayor que los polinomios factores (a lo más de grado $2n-2$) y eso no está bien porque se saldría del campo en el que estamos trabajando. Es por ello que necesitamos, como en el caso de los campos primos, trabajar módulo alguien que garantice la cerradura y para asegurar que todo elemento del campo tenga inverso se requiere entonces que el módulo sea un elemento no factorizable, es decir, un polinomio irreducible.

Ejemplo 4.2.1. Sea $P(x) = x^3 + x + 1$.

$$(x^2+x+1) \times (x^2+1) \pmod{P(x)} = x^4+x^3+x^2+x^2+x+1 \pmod{P(x)}$$

Como:

$$\begin{array}{rr|l} & & x+1 \\ \cline{3-3} x^3+x+1 & x^4+x^3 & +x+1 \\ & -x^4 & -x^2-x \\ \cline{2-3} & x^3-x^2 & +1 \\ & -x^3 & -x-1 \\ \cline{2-3} & -x^2-x & \end{array}$$

Entonces:

$$(x^2 + x + 1) \times (x^2 + 1) \pmod{P(x)} = -x^2 - x = x^2 + x$$

Por cierto $x^3 + x + 1$ es un polinomio irreducible de grado 3, así que debe existir Q, tal que:

$$(x^2 + x + 1) \times Q \pmod{P(x)} = 1$$

En efecto

$$(x^2 + x + 1) \times x^2 \pmod{P(x)} = x^4 + x^3 + x^2 \pmod{P(x)}$$

Además:

$$\begin{array}{r|l} & x + 1 \\ \cline{2-2} x^3 + x + 1 & x^4 + x^3 + x^2 \\ & \underline{-x^4 \qquad -x^2 \quad -x} \\ & \qquad\quad x^3 \qquad -x \\ & \qquad \underline{-x^3 \qquad -x - 1} \\ & \qquad\qquad\quad -2x - 1 \end{array}$$

De donde:

$$(x^2 + x + 1) \times x^2 \pmod{P(x)} = 1$$

$\triangleleft$

Ejemplo 4.2.2. Usando los polinomios del otro ejemplo y usando como polinomio irreducible:

$$AES(x) = x^8 + x^4 + x^3 + x + 1 \tag{4.1}$$

$$\begin{aligned} (x^7 + x^6 + x^4) \times (x^4 + x^2 + 1) \pmod{AES(x)} \\ = x^{11} + x^{10} + x^9 + 2x^8 + x^7 + 2x^6 + x^4 \\ = x^{11} + x^{10} + x^9 + x^7 + x^4 \\ = x^4 + x \end{aligned}$$

Dado que:

$$
\begin{array}{r|l}
 & \hphantom{x^{11} + x^{10} + x^9 + x^7 + x^4 - x^3 - x^6} x^3 + x^2 + x \\
\cline{2-2}
x^8 + x^4 + x^3 + x + 1 & x^{11} + x^{10} + x^9 + x^7 + x^4 \\
 & - x^{11} - x^7 - x^6 - x^4 - x^3 \\
\cline{2-2}
 & \quad x^{10} + x^9 - x^6 - x^3 \\
 & \quad - x^{10} - x^6 - x^5 - x^3 - x^2 \\
\cline{2-2}
 & \qquad x^9 - 2x^6 - x^5 - 2x^3 - x^2 \\
 & \qquad - x^9 - x^5 - x^4 - x^2 - x \\
\cline{2-2}
 & \qquad\qquad - 2x^6 - 2x^5 - x^4 - 2x^3 - 2x^2 - x
\end{array}
$$

◁

Por supuesto la propiedad de ser irreducible no la posee cualquier polinomio. Por ejemplo: $x^4 + x^3 + x + 1$ no lo es ya que:

$$x^4 + x^3 + x + 1 = (x^2 + x + 1)(x^2 + 1)$$

Hay que hacer notar que en el campo finito $GF(2^n)$ los elementos se representan como polinomios de grado menor a n y que, sin embargo, el polinomio irreducible que se usa como módulo para operar, es de grado n, es decir, *no* es un elemento del campo $GF(2^n)$. En nuestro ejemplo previo, el polinomio $AES(x) = x^8 + x^4 + x^3 + x + 1$ se puede usar como el módulo para trabajar en $GF(2^8)$, donde todos los elementos tienen grado máximo de 7. Por cierto no es el único que se puede usar, también: $Q(x) = x^8 + x^4 + x^3 + x^2 + 1$, es irreducible.

Ejemplo 4.2.3. Usando los mismos polinomios del ejemplo previo y usando ahora el polinomio irreducible $Q(x) = x^8 + x^4 + x^3 + x^2 + 1$:

$$
\begin{aligned}
(x^7 + x^6 + x^4) \times (x^4 + x^2 + 1) \pmod{Q(x)} \\
= x^{11} + x^{10} + x^9 + 2x^8 + x^7 + 2x^6 + x^4 \\
= x^{11} + x^{10} + x^9 + x^7 + x^4 \\
= x^5 + x^4 + x^2 + x
\end{aligned}
$$

Dado que:

$$
\begin{array}{r|l}
 & x^3+x^2+x \\
\hline
x^8+x^4+x^3+x^2+1 & \begin{array}{rrrrrrr}
x^{11}+x^{10}+x^9+x^7 & & & +x^4 & & & \\
-x^{11} & & -x^7 & -x^6 & -x^5 & & -x^3 \\
\hline
x^{10}+x^9 & & & -x^6 & -x^5+x^4 & & -x^3 \\
-x^{10} & & & -x^6 & -x^5-x^4 & & -x^2 \\
\hline
x^9 & & -2x^6 & -2x^5 & & -x^3 & -x^2 \\
-x^9 & & & & -x^5-x^4 & -x^3 & -x \\
\hline
 & & -2x^6 & -3x^5 & -x^4 & -2x^3 & -x^2-x
\end{array}
\end{array}
$$

$\lhd$

El resultado es ahora muy diferente del previo. Sin embargo $GF(2^8)$ es único salvo isomorfismo, así que lo que hacemos al cambiar el módulo es realmente reetiquetar todos los elementos del campo, cambiarles de nombre.

Es sobre todo en el contexto de las implementaciones en dispositivos de cómputo, que los elementos del campo finito, los polinomios, se codifican en binario o, para abreviar, como números en base 8 o 16. Así, por ejemplo, en $GF(2^3)$ los elementos se pueden representar como polinomios de grado menor o igual a 2 sobre $GF(2)$, es decir con coeficientes binarios; así que cualquier elemento del campo puede codificarse usando el valor de sus tres coeficientes. El polinomio $x^2+1 = 1\,x^2+0\,x+1$ se puede codificar como 101, y $x+1$ como 011.

Ejemplo 4.2.4. Trabajando en $GF(2^4)$ usando $M(x) = x^4+x+1$ como módulo.

Sea $A(x) = x^3+x^2+1$ y $B(x) = x^2+x$. El código de A es $1101_2 = 13 = \texttt{D}$, el de B es $0110_2 = 6$ (véase la tabla 4.3).

Para abreviar más adelante, conviene hacer de una vez:

$$
\begin{aligned}
x^5 &= x\,M(x) + (x^2+x) \pmod{M(x)} \\
&= x^2+x
\end{aligned} \tag{4.2}
$$

así que

$$
\begin{aligned}
C'(x) &= A(x)\,B(x) \\
&= x^5+x^4+x^2+x^4+x^3+x \\
&= x^5+x^3+x^2+x
\end{aligned}
$$

Ahora en $GF(2^4)$, usando 4.2:

$$\begin{aligned} C(x) &= C'(x) \pmod{M(x)} \\ &= x^2 + x + x^3 + x^2 + x \pmod{M(x)} \\ &= x^3 \end{aligned} \tag{4.3}$$

En síntesis: $(x^3 + x^2 + 1)\ (x^2 + x) = x^3 \pmod{M(x)}$
El código de x^3 es $1000_2 = 8$ así que podemos escribir que: $D \times 6 = 8$

◁

4.3. Inverso multiplicativo

Además de la suma, la diferencia y el producto también tenemos, claro está, la existencia de un inverso multiplicativo. El inverso de un elemento en el campo puede hallarse usando el algoritmo extendido de Euclides. Dado que las operaciones se llevan a cabo módulo un polinomio irreducible $S(x)$, calcular el inverso de un elemento $A(x)$ consiste en calcular el máximo común divisor (MCD) de $A(x)$ y $S(x)$, lo que sabemos que será 1. El algoritmo extendido de Euclides nos proporciona el MCD escrito como combinación lineal de A y S, así que tendremos algo como:

$$1 = Q(x)\, S(x) + R(x)\, A(x) \pmod{S(x)}$$

Por supuesto el primer sumando se anula al ser múltiplo de S y lo que nos queda es:

$$R(x)\, A(x) = 1$$

lo que significa que R es el polinomio inverso multiplicativo de A en el campo finito en que trabajamos.

Ejemplo 4.3.1. En $GF(2^8)$ con módulo $AES(x) = x^8 + x^4 + x^3 + x + 1$
Sea $A(x) = x^7 + x^6 + x = \texttt{1100,0010} = \texttt{C2}$
El algoritmo extendido de Euclides nos dice que:

$$\begin{aligned} Q(x) &= x^4 + x^3 + x^2 + 1 \\ R(x) &= x^5 + x^3 + x^2 + x + 1 \end{aligned}$$

de donde: $A^{-1} = R(x) = x^5 + x^3 + x^2 + x + 1 = \texttt{0010,1111} = \texttt{2F}$

◁

Ejemplo 4.3.2. En $GF(2^4)$ con módulo $M(x) = x^4 + x + 1$. Calcularemos el inverso multiplicativo de $x^2 + 1 = \texttt{0101}_2 = \texttt{5}$.

$$\begin{array}{rr|l} & x^2 & -1 \\ \hline x^2+1 & x^4 & +x+1 \\ & -x^4-x^2 & \\ \hline & -x^2+x+1 & \\ & x^2 & +1 \\ \hline & x+2 & \end{array}$$

Considerando que los coeficientes están en $GF(2)$:

$$x^4 + x + 1 = (x^2+1)(x^2+1) + x$$

O bien:

$$x = (x^4 + x + 1) - (x^2+1)(x^2+1) \tag{4.4}$$

Ahora:

$$\begin{array}{rr|l} & & x \\ \hline x & x^2 & +1 \\ & -x^2 & \\ \hline \end{array}$$

Es decir:

$$x^2 + 1 = x \cdot x + 1$$

O bien:

$$1 = (x^2+1) - x \cdot x \tag{4.5}$$

Usando 4.4 en 4.5:

$$\begin{aligned} 1 &= (x^2+1) - x[(x^4+x+1) - (x^2+1)(x^2+1)] \pmod{M(x)} \\ &= -x(x^4+x+1) + x(x^2+1)(x^2+1) + (x^2+1) \pmod{M(x)} \\ &= x(x^4+x+1) + (x^2+1)(x^3+x) + (x^2+1) \pmod{M(x)} \\ &= x(x^4+x+1) + (x^2+1)(x^3+x+1) \pmod{M(x)} \end{aligned}$$

De donde: $(x^2+1)^{-1} = x^3 + x + 1 = \texttt{1011}_2 = \texttt{B}$

En la tabla 4.3 aparecen los inversos de todos los elementos de $GF(2^4)$ usando a $M(x)$ como módulo.

$\lhd$

Polinomio	**Bin.**	**Hex.**	**Inv.**
1	0001	1	1
x	0010	2	9
$x+1$	0011	3	E
x^2	0100	4	D
x^2+1	0101	5	B
x^2+x	0110	6	7
x^2+x+1	0111	7	6
x^3	1000	8	F
x^3+1	1001	9	2
x^3+x	1010	A	C
x^3+x+1	1011	B	5
x^3+x^2	1100	C	A
x^3+x^2+1	1101	D	4
x^3+x^2+x	1110	E	3
x^3+x^2+x+1	1111	F	8

Tabla 4.1: Inversos en $GF(2^4)$ usando $M(x) = x^4 + x + 1$ como módulo.

Polinomio	Bin.	Hex.	Inv.
1	0001	1	1
x	0010	2	C
$x+1$	0011	3	8
x^2	0100	4	6
x^2+1	0101	5	F
x^2+x	0110	6	4
x^2+x+1	0111	7	E
x^3	1000	8	3
x^3+1	1001	9	D
x^3+x	1010	A	B
x^3+x+1	1011	B	A
x^3+x^2	1100	C	2
x^3+x^2+1	1101	D	9
x^3+x^2+x	1110	E	7
x^3+x^2+x+1	1111	F	5

Tabla 4.2: Inversos en $GF(2^4)$ usando $M'(x) = x^4 + x^3 + 1$ como módulo.

Ejemplo 4.3.3. Ahora usando el mismo $GF(2^4)$ pero con $M'(x) = x^4 + x^3 + 1$ como módulo.

Si $A(x) = x^3 + x^2 + 1$ y $B(x) = x^2 + x + 1$ entonces:

$$A(x)\,B(x) = x^3 \pmod{M'(x)}$$

En este caso, el inverso de $x^2 + 1 = \texttt{0101}_2 = \texttt{5}$ es:

$$x^3 + x^2 + x + 1 = \texttt{1111}_2 = \texttt{F}$$

$\triangleleft$

En los ejemplos hemos usado dos polinomios diferentes como el módulo para obtener $GF(2^4)$. Pero sabemos que éste, como todos los campos finitos, realmente son únicos, así que las dos representaciones, usando M o M' deben ser isomorfas, es decir mapeables entre sí biyectivamente. En efecto, en la tabla 4.3 se hace explícito el mapeo. Sea $f : GF(2^4)_M \mapsto GF(2^4)_{M'}$ el isomorfismo explícito en la tabla 4.3. Si p y q son polinomios en la representación $GF(2^4)_M$ y $f(p) = p'$ y $f(q) = q'$ los respectivos resultados de mapearlos en $GF(2^4)_{M'}$, entonces debe ocurrir que $f(p \cdot q) = f(p) \cdot f(q) = p' \cdot q'$

Ejemplo 4.3.4. Sean $p = x + 1$ y $q = x^2 + 1$ en $GF(2^4)_M$.

En $GF(2^4)_M$, o sea usando $M(x) = x^4 + x + 1$ como módulo:

$$p \cdot q = x^3 + x^2 + x + 1 = r$$

De acuerdo con el isomorfismo de la tabla 4.3: $p' = f(x + 1) = x^3 + 1$ y $q' = f(x^2 + 1) = x^3 + x^2 + x$.

Así que por otra parte, en $GF(2^4)_{M'}$, usando $M'(x) = x^4 + x^3 + 1$ como módulo:

$$p' \cdot q' = x^6 + x^5 + x^4 + x^3 + x^2 + x \pmod{M'(x)}$$

Por cierto:

$$\begin{array}{r|l} & \quad\quad\quad\quad x^2 \quad\quad + 1 \\ \hline x^4 + x^3 + 1 & \;\; x^6 + x^5 + x^4 + x^3 + x^2 + x \\ & - x^6 - x^5 \quad\quad\quad\quad - x^2 \\ \hline & \quad\quad\quad\quad x^4 + x^3 \quad\quad + x \\ & \quad\quad\quad\quad - x^4 - x^3 \quad\quad\quad - 1 \\ \hline & \quad\quad\quad\quad\quad\quad\quad\quad x - 1 \end{array}$$

De donde:

$$p' \cdot q' = x + 1 = r'$$

Se verifica que: $r' = f(r)$

$\lhd$

$M(x)$			$M'(x)$		
Polinomio	**Bin.**	**Hex.**	**Hex.**	**Bin.**	**Polinomio**
1	0001	1	1	0001	1
x	0010	2	2	0010	x
$x+1$	0011	3	9	1001	x^3+1
x^2	0100	4	4	0100	x^2
x^2+1	0101	5	E	1110	x^3+x^2+x
x^2+x	0110	6	B	1011	x^3+x+1
x^2+x+1	0111	7	A	1010	x^3+x
x^3	1000	8	8	1000	x^3
x^3+1	1001	9	C	1100	x^3+x^2
x^3+x	1010	A	5	0101	x^2+1
x^3+x+1	1011	B	7	0111	x^2+x+1
x^3+x^2	1100	C	F	1111	x^3+x^2+x+1
x^3+x^2+1	1101	D	6	0110	x^2+x
x^3+x^2+x	1110	E	D	1101	x^3+x^2+1
x^3+x^2+x+1	1111	F	3	0011	$x+1$

Tabla 4.3: Isomorfismo entre las dos representaciones de $GF(2^4)$ usando $M(x) = x^4 + x + 1$ y $M'(x) = x^4 + x^3 + 1$ como módulo respectivamente.

5. El caso de AES

Lo que hemos revisado hasta ahora forma parte del sustento teórico del Estándar de Cifrado Avanzado o *Advanced Encryption Standard* (AES) del Instituto Nacional de Estándares y Tecnología Norteamericano. De hecho es por ello que hemos llamado así al polinomio 4.1.

AES recibe como entrada dos números que, al estar almacenados en la memoria de un dispositivo de cómputo digital, supondremos representados en binario.

- Una clave K que puede medir 128, 192 o 256 bits de longitud.
- Un bloque de datos D a cifrar de 128 bits de longitud.

El bloque D se almacena byte[1] por byte en una matriz de 4×4 a la que en la documentación del estándar se le denomina el *estado* y que denotaremos como E.

La operación de AES se divide en rondas cuyo número depende del tamaño de clave elegido y pueden ser 10, 12 o 14 para los respectivos tamaños de clave mencionados arriba. Todas las rondas, salvo la última, son idénticas y su labor se puede, a su vez, dividir en tres etapas:

1. `SubBytes`.
2. `ShiftRows`.
3. `MixColumns`.

La única diferencia entre la última ronda y las previas es que en ella no se aplica el algoritmo de `MixColumns`.

[1] Como es usual un byte son 8 bits consecutivos.

`SubBytes`

1. Se considera cada byte E_i de la matriz de estado, como la representación de un polinomio en $GF(2^8)$ que hemos mencionado, usando el polinomio AES de la expresión 4.1 como módulo.

2. Se obtiene el inverso multiplicativo, E_i^{-1} de cada uno de los 16 polinomios en la matriz de estado.

3. Se aplica a cada uno de los polinomios inversos una transformación afín fija, común a todos ellos: $A(E_i^{-1}) = M\,E_i^{-1} + V \pmod 2$ donde:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{5.1}$$

$$V = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{5.2}$$

   O equivalentemente, si llamamos e'_j al j-ésimo bit de E_i^{-1}, a_j al respectivo de $A(E_i^{-1})$ y v_j al de V:

$$a_j = e'_j \oplus e'_{(j+4)\bmod(8)} \oplus e'_{(j+5)\bmod(8)} \oplus e'_{(j+6)\bmod(8)} \oplus e'_{(j+7)\bmod(8)} \oplus v_j$$

El polinomio usado como módulo: $AES(x) = x^8 + x^4 + x^3 + x + 1$, es irreducible, como ya sabemos, pero no primitivo. Los autores del cifrado prefirieron elegir este polinomio porque además de que no era necesario que fuera primitivo, los polinomios primitivos generaban patrones más claros, eran más lineales a la hora de obtener el inverso de un elemento, cosa que es

indeseable en un cifrado. De hecho es la obtención del inverso la operación que concede no-linealidad a AES, la que concede, en términos de Shannon, la *confusión* al sistema.

Por supuesto en las implementaciones de AES no se calcula el inverso de cada byte/polinomio, en aras de la eficiencia todos los 256 inversos[2] son precalculados y almacenados en una tabla. Del mismo modo los 256 posibles resultados de la transformación afín se buscan en una tabla de resultados y no se calculan al momento.

`ShiftRows`

En la etapa de `ShiftRows`, como el nombre lo indica, se desplazan los renglones de la matriz de estado. La dirección del desplazamiento es siempre un número entero de bytes a la izquierda. Los bytes que "salen" por la izquierda regresan en orden pr la derecha, así que técnicamente lo que se lleva a cabo es una rotación. El tamaño de la rotación es diferente para cada renglón.

Renglón 0 Se desplaza cero bytes, es decir no se mueve en absoluto.

Renglón 1 1 byte.

Renglón 2 2 bytes.

Renglón 3 3 bytes.

El propósito de esta operación es la difusión.

`MixColumns`

Esta es la otra operación que tiene interés en el contexto de nuestro tema. Para aplicarla se sigue pensando cada byte de la matriz de estado como un elemento de $GF(2^8)$, pero ahora además los cuatro bytes de cada columna se consideran a su vez, como los coeficientes de un polinomio de grado menor a 4.

Denotaremos con $C_i(x)$ al polinomio asociado a la i-ésima columna de la matriz de estado y a c_j como el j-ésimo de sus coeficientes. Las operaciones con los polinomios de las columnas se harán módulo $M(x) = x^4 + 1$. Sea $P(x)$ el polinomio:

$$P(x) = 3x^3 + x^2 + x + 2 \tag{5.3}$$

[2]Por supuesto el cero no tiene inverso, sin embargo en la tabla de inversos de AES el cero se mapea en sí mismo.

la operación `MixColumns` consiste en obtener:

$$R(x) = C_i(x)\, P(x) \pmod{M(x)}$$

Como sabemos esto es también equivalente a hacer lo siguiente:

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

El polinomio usado como módulo $M(x) = x^4 + 1$ no es irreducible, pero se lo utiliza porque no tiene divisores comunes con el polinomio 5.3, por lo que éste es entonces invertible módulo M, lo que hace factible invertir la operación `MixColumns` al descifrar.

En términos de Shannon, el objetivo de `MixColumns` es la difusión de los datos: lograr que cada bit de la entrada influya en tantos bits de resultado como sea posible. De hecho en AES luego de sólo 3 rondas, cada byte de salida depende de los 16 bytes de entrada.

Apéndice: código

El siguiente es un pequeño programa escrito para el software SageMath[3] que lista los elementos del campo finito $GF(2^4)$ con el módulo $x^4 + x + 1$ y sus inversos. Se puede usar el mismo programa para encontrar los inversos en mismo campo usando el generador $x^4 + x^3 + 1$ modificando la línea 2 del código, donde se especifica el generador (usando b en vez de x).

```
1 F.<b> = GF(2)[]
2 S.<x> = GF(2^4, modulus = b^4 + b + 1)
3 for i in range(S.rank(1)):
4      x = S.unrank(i + 1)
5      nv = 1/x
6      print("{}\t{}\t\t␣{}".format(i + 1, x, nv))
```

Para imprimir la tabla de multiplicar se puede usar algo similar al siguiente código:

```
1 F.<b> = GF(2)[]
2 S.<x> = GF(2^4, modulus = b^4 + b + 1)
3 for i in range(S.rank(1)):
4          x = S.unrank(i + 1)
5          for j in range(S.rank(1)):
6              y = S.unrank(j + 1)
7              pr = x * y
8              print("{}\t".format(pr)),
9          print("\n")
```

[3]Disponible con licencia de software libre en: `http://www.sagemath.org/`

Para generar la tabla que define el isomorfismo usado como ejemplo se puede usar el siguiente código[4].

```
F.<b> = GF(2)[]
F1.<x> = GF( 2^4, modulus = b^4 + b + 1 )
F2.<x> = GF( 2^4, modulus = b^4 + b^3 + 1 )
tabla1 = F1.multiplication_table(names='elements')
tabla2 = F2.multiplication_table(names='elements')
zip(tabla1.column_keys(), tabla2.column_keys())
```

`F1` es $GF(2^4)$ usando x^4+x+1 como módulo, mientras que `F2` representa al mismo campo usando $x^4 + x^3 + 1$ como módulo.

[4]Inspirado en código original de Edgar Arroyo, ayudante del curso de criptografía que imparte el autor.

Bibliografía

[1] NIST, *Announcing the Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication (FIPS) 197. United States National Institute of Standards and Technology (NIST). 26 de noviembre de 2001.

[2] Lidl, R. y H. Niederreiter, *Introduction to finite fields and their applications*, Cambridge University Press, 1986.

[3] Paar, C. y Pelzl J., *Understanding Cryptography*, Springer, 2010.

[4] Rotman, J., *Galois theory*, 2a Ed., Springer-Verlag, 1998.

[5] Sage Development Team, *SAGE Tutorial*, Ver. 8.1, 2017. Disponible en: `http://doc.sagemath.org/pdf/en/tutorial/SageTutorial.pdf`

[6] Yanajara, Parra H., (traductor) *Manual de SAGE para principiantes*, Instituto Tecnológico de Sonora, 2010. Disponible en: `http://www.sagemath.org/es/Manual_SAGE_principiantes.pdf`