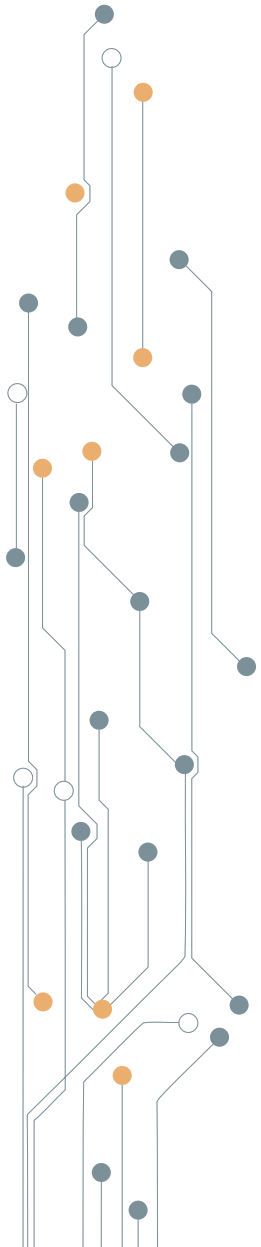
Aritmética modular para la criptografía

Telefónica EDUCACIÓN DIGITAL

Índice



1. El concepto de módulo

En criptografía las operaciones de cifra se realizan dentro de un cuerpo de cifra o módulo, utilizando aritmética modular.

¿Qué es la aritmética modular? De la misma manera que la manecilla del segundero de un reloj al llegar a los 60 segundos se posiciona en el valor inicial 0 y en el minutero se apunta que ha transcurrido otro minuto, lo que obviamente puede aplicarse también al minutero y al horario, en matemática discreta decimos que un cuerpo finito n está conformado por n números enteros, que van desde el valor 0 hasta el valor n-1. Decimos entonces que trabajamos en un módulo n cuando los elementos son todos los números enteros que van desde 0 hasta n-1. Dichos números se conocen como restos.

Figura 2.1. Operación módulo 60 en el segundero y el minutero, y módulo 12 en el horario.

Esto es, los restos del número n igual a 7 serán {0, 1, 2, 3, 4, 5, 6} y los restos del número n igual a 10 serán {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Observa que si el número n es primo (el caso del 7), todos sus restos {1, 2, 3, 4, 5, 6} (el cero no se tiene en cuenta) son primos relativos con él, y si el número n es compuesto, como es el caso del 10, sólo algunos restos serán primos relativos con él {1, 3, 7, 9}. Esto que parece no tener aquí ninguna importancia por ser algo obvio, sí tendrá trascendencia en la cifra cuando hablemos de los inversos, es decir aquellos números que nos permitirán deshacer una operación de cifra, o lo que es lo mismo, descifrar.

Como en el ejemplo del reloj, si tras una operación el número resultante es igual o superior a n, o bien inferior a n, entraremos en lo que se conoce como grados de equivalencia. Por ejemplo, son equivalentes al resto 5 en mod 27 entre una infinidad de valores los números 32, 140 y -76, pues 32 = 27x1 + 5; 140 = 27x5 + 5; y -76 = 27x(-3) + 5. De la misma forma, 189 mod 27 = 0 porque 189 = 27x7 + 0.

2. Operaciones modulares en criptografía

Las operaciones modulares que se realizan en los algoritmos criptográficos son, básicamente, estas cinco: la reducción a módulo, la suma dentro del cuerpo, la suma módulo 2 también conocida como or exclusivo o xor, la multiplicación y la elevación a potencia o exponenciación.

En la tabla 2.1 se incluyen algunos ejemplos de estas operaciones modulares dentro del módulo n = 60. Los resultados siempre estarán comprendidos entre el 0 y el n-1, en este caso el 59. No obstante, el valor que va a reducirse módulo n puede ser cualquiera, como se aprecia en la tabla 2.1, e.g. 239 mod 60.

Reducción a módulo n	45 mod 60 = 45	120 mod 60 = 0	239 mod 60 = 59	-15 mod 60 = 45
Suma módulo n	38 +90 mod 60 = 8	31+150 mod 60 = 1	40 + 80 mod 60 = 0	25 – 50 mod 60 = 35
Suma módulo 2 or exclusivo	2 + 2 = 0	2 + 3 = 1	2 + 4 = 6	2 + 5 = 7
Multiplicación módulo n	13*6 mod 60 = 18	5*60 mod 60 = 0	11*11 mod 60 = 1	-3*50 mod 60 = 30
Exponenciación módulo n	24 mod 60 = 16	34 mod 60 = 21	559 mod 60 = 5	105 mod 60 = 40

Tabla 2.1. Ejemplos de operaciones modulares dentro del cuerpo n = 60.

Exponenciación rápida

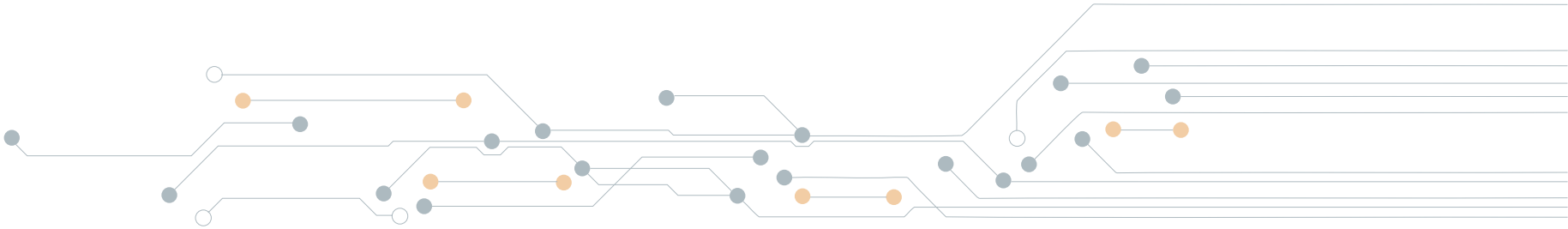
Una operación muy común en criptografía moderna es la de elevar a potencia o exponenciación dentro de un cuerpo, de la forma AB mod n. Si los números A, B y n son pequeños, por ejemplo 312 mod 121, perfectamente podríamos resolver primero la potencia 312 = 531.441 y luego reducir este valor a módulo 121, para obtener el resultado 9.

Pero, ¿qué sucedería si estos valores son muy grandes, de centenas e incluso miles de bits? No podemos ahora resolver primero la potencia porque, entre otras cosas, no contamos con suficiente espacio para guardar un número tan grande, por ejemplo, 97.7138.701 mod 1.000.003. Aquí, aunque el módulo sea un valor muy pequeño (sólo 7 dígitos), la potencia 97.7138.701 será un número de 43.417 dígitos.

Puesto que números A, B y n, de 5, 4 y 7 dígitos respectivamente, usados en la ecuación anterior, son extremadamente pequeños comparados con los que se usan en la criptografía actual, con módulos de dos mil bits (unos 600 dígitos), resulta claro que debemos aplicar algunas propiedades de la aritmética discreta para permitir realizar esta operación.

Uno de los métodos para resolver AB mod n = x es el siguiente:

- Representar el exponente B en formato binario de k bits: bk-1bk-2... bi...b1b0
- Hacer x = 1
- Para i = k-1,... 0 hacer
  - x = x2 mod n
  - Si (bi = 1) entonces
  - x = x*A mod n



Es decir, primero convertimos el exponente B a su valor binario. A continuación, en módulo n haremos operaciones consistentes en elevar al cuadrado restos de ese módulo y, en algunos casos y cuando corresponda, multiplicaremos además ese valor por la base A. Como estas operaciones se hacen desde i = k-1 hasta k = 0, solamente realizaremos k operaciones, siendo k el número de bits del exponente, reduciendo drásticamente el total de operaciones comparado con el método de los cuadrados que se expondrá después del ejemplo. La tabla 1.2 muestra las operaciones que deben realizarse para encontrar el resultado de 25137 mod 501 = 211.

$137 = 10001001 = b_7b_6b_5b_4b_3b_2b_1b_0$		
$b_7 = 1$	$x = 1^2 * 25 \bmod 501$	$x = 25$
$b_6 = 0$	$x = 25^2 \bmod 501$	$x = 124$
$b_5 = 0$	$x = 124^2 \bmod 501$	$x = 346$
$b_4 = 0$	$x = 346^2 \bmod 501$	$x = 478$
$b_3 = 1$	$x = 478^2 * 25 \bmod 501$	$x = 199$
$b_2 = 0$	$x = 199^2 \bmod 501$	$x = 22$
$b_1 = 0$	$x = 22^2 \bmod 501$	$x = 484$
$b_0 = 1$	$x = 484^2 * 25 \bmod 501$	$x = 211$

Tabla 2.2. Ejemplo del método de exponenciación rápida

El método de los cuadrados habría significado 68 (=136/2) operaciones del tipo (252 mod 501), las 67 multiplicaciones entre esos valores intermedios reducidos módulo 501, y una última multiplicación por 25 módulo 501 (porque el exponente es impar):

$25^{137} \bmod 501 = (25^2 \bmod 501 * 25^2 \bmod 501 * 25^2 \bmod 501 *$

... *...

***$25^2 \bmod 501 * 25^2 \bmod 501 * 25^2 \bmod 501) * 25 \bmod 501$**

Es decir, 136 operaciones. Esto es, muchas más que las 9 del método rápido. Como los números A, B y n en la práctica son muchísimo mayor, el alto rendimiento de este método rápido queda muy claro; serán miles de millones de operaciones menos.

3. Inversos en un cuerpo

¿Qué son los inversos en un cuerpo? Como en la cifra realizaremos en emisión operaciones de suma y multiplicación (o exponenciación) módulo n de los elementos del texto en claro con números o alguna clave, o también suma or exclusivo módulo 2 en algunos algoritmos de cifra moderna, debemos asegurarnos de que en el extremo receptor se pueda deshacer dicha operación, esto es descifrar, y recuperar así el texto en claro.

Para ello usaremos los inversos, de manera que en el extremo receptor el destinatario de la cifra, conociendo generalmente una clave, realizará la misma operación que en el extremo emisor, pero aplicando los valores inversos a los usados en emisión.

Inverso aditivo

Si en la operación de cifra un algoritmo ha utilizado como clave una suma módulo n, en recepción se podrá descifrar el criptograma recibido usando el mismo algoritmo, pero con el inverso de esa clave.

El inverso aditivo de un número a en un cuerpo n, que denotaremos por inv+ (a, n), será el complemento a dicho cuerpo, en tanto se obtiene la identidad de la suma.

Es decir, si en módulo 13 aplicamos como clave una suma o desplazamiento del código o mensaje a cifrar de 5 posiciones hacia la derecha, en recepción habrá que aplicar ese mismo desplazamiento al criptograma, pero en sentido contrario, hacia la izquierda, es decir -5. Pero esto es lo mismo que aplicar un desplazamiento en el mismo sentido de la cifra, pero con el complemento de 5 en módulo 13, que es igual a 8, dado que 5 + 8 mod 13 = 0. A continuación, tres ejemplos:

inv+ (18, 27) = 9	inv+ (110, 1000) = 890	inv+ (-15, 51) = 11 Primero se resuelve -15 mod 51 = 36 y después inv+ (36,51) = 15

Tabla 2.3. Ejemplos de inversos aditivos.

Inverso or exclusivo o módulo 2

Un caso especial de inversos aditivos lo tenemos en la operación XOR y que se usará solamente en la cifra moderna. La operación XOR equivale a una suma módulo 2, es decir usando como restos el 0 y el 1; por lo tanto, aplicable sólo al entorno de operaciones digitales.

La tabla de verdad de la operación or exclusivo indica que la operación entre bits iguales entrega como resultado un 0 y la operación entre bits diferentes entregas como resultado un 1. Así, si sumamos or exclusivo el número 2 con el número 3 el resultado será: 10 11 = 01 = 1.

El inverso de un or exclusivo será el mismo valor.

Por ejemplo, si la letra Z (en ASCII = 90 = 1011010) se cifra xor con la clave s (en ASCII = 115 = 1110011), el resultado será:

- Z s = 90 115 = 1011010 1110011 = 101001 = 41 que es el carácter).

Si ahora desciframos el criptograma) con la clave s, recuperamos el texto en claro:

- 101001 1110011 = 1011010 = 90 = Z

4. Inversos multiplicativos

Dedicamos un apartado sólo para inversos multiplicativos porque son los más importantes y más utilizados en la criptografía moderna.

Se dice que un número a, elemento o resto del cuerpo n, tiene inverso multiplicativo en dicho cuerpo, o simplemente inverso, si existe otro número x que haga cumplir la condición de que a * x mod n = 1.

Es decir, que al multiplicar el valor de a por el valor de x y reducir el resultado módulo n, se obtenga el valor 1, la identidad de la multiplicación.

De la ecuación anterior, y aunque matemáticamente no sea correcto decirlo, por simplicidad y para que podamos recordarlo de forma nemotécnica, podríamos asociar el concepto del inverso a que a = 1/x o bien que x = 1/a en ese cuerpo n. En aritmética modular no es lo mismo decir que x = 1/a mod n (lo cual no está permitido) a que x = a-1 mod n, o lo que es lo mismo, x = inv (a, n), que sí es una expresión correcta.

Para que un número a dentro de un módulo n tenga inverso multiplicativo, deberá cumplirse que mcd (a, n) =1.

Como ejercicio, veamos cuál será inverso de 2 en módulo 15, multiplicando a = 2 por todos los restos de cuerpo n = 15.

2 x 0 mod 15 = 0	2 x 1 mod 15 = 2	2 x 2 mod 15 = 4
2 x 3 mod 15 = 6	2 x 4 mod 15 = 8	2 x 5 mod 15 = 10
2 x 6 mod 12 = 12	2 x 7 mod 15 = 14	2 x 8 mod 15 = 1
2 x 9 mod 15 = 3	2 x 10 mod 15 = 5	2 x 11 mod 12 = 7
2 x 12 mod 15 = 9	2 x 13 mod 15 = 11	2 x 14 mod 15 = 13

Tabla 2.4. Buscando el inverso de 2 en módulo 15

El inverso existe porque mcd (2, 15) = 1, dado que 15 = 3x5. En la tabla se observa que las 15 operaciones han entregado resultados distintos y estos forman todo el cuerpo de cifra, los restos que van desde 0 hasta n-1 =14. Y sólo una operación ha entregado el valor buscado de la unidad, 2 x 8 mod 15 = 1. Por lo tanto, el inverso de 2 en módulo 15 es 8, así como el inverso de 8 en módulo 15 será 2.

inv (2, 15) = 8; inv (8, 15) = 2

Por lo tanto, si ahora se cifra el número secreto 10 en el cuerpo n = 15 con la clave a = 2, se obtiene:

- 10 x 2 mod 15 = 5

Y si ahora desciframos el criptograma 5 con el inv (2, 15) = 8; es decir:

- 5 x 8 mod 15 = 10 (recuperamos el secreto)

Observa que es distinto multiplicar por el inverso de la clave que dividir por la clave (lo que no está permitido), porque si el criptograma 5 lo dividimos por la clave a = 2, obtendríamos 2,5 lo que no es válido ya que en aritmética modular se trabaja sólo con enteros.

Algoritmo Extendido de Euclides

Lógicamente, no podremos usar el método de la tabla 4.1 para encontrar inversos con números muy grandes, como los que se usan habitualmente en la criptografía moderna, sería totalmente impracticable.

Para ello existe un método basado en el algoritmo de Euclides, conocido como algoritmo extendido de Euclides para calcular x = inv (a, n) y que se muestra a continuación.

Hacer (g_0, g_1, u_0, u_1, v_0, v_1, i) = (n, a, 1, 0, 0, 1, 1)

Mientras gi distinto 0 hacer

  Hacer y_{i+1} = parte entera (g_{i-1}/g_i)

  Hacer $g_{i+1} = g_{i-1} - y_{i+1} \times g_i$

  Hacer $u_{i+1} = u_{i-1} - y_{i+1} \times u_i$

  Hacer $v_{i+1} = v_{i-1} - y_{i+1} \times v_i$

  Hacer i = i+1

Si (vi-1 < 0)

  Hacer $v_{i-1} = v_{i-1} + n$

  Hacer $x = v_{i-1}$

La figura 2.2 muestra el desarrollo del algoritmo para encontrar el inv (9, 275).

i	y_i	g_i	u_i	v_i
0	-	275	1	0
1	-	9	0	1
2	30	5	1	-30
3	1	4	-1	-30
4	1	1	2	-61
5	4	0	-9	275

Figura 2.2. Algoritmo extendido de Euclides para el cálculo del inv (9, 275).

Como v_{i-1} = -61, es decir ha salido un valor negativo, entonces X = - 61 + 275 = 214. Efectivamente, inv (9, 275) = 214 pues 214 x 9 = 1.926 mod 275 = 1, puesto que 1.926 = 7 x 275 + 1.

5. Problemas P y NP en criptografía

En matemáticas hay un conjunto de funciones que, al operar con ellas en un sentido, las operaciones son fáciles de realizar y muy rápidas. Esto es, que significan un número de operaciones bit en una máquina determinista (máquina de Turing) que están en proporción con el tamaño de los datos de entrada. Por lo tanto, el problema puede ser resuelto en esa máquina en un período de tiempo polinómico (P), lo cual podríamos asociar a un funcionamiento lineal del tiempo de respuesta.

Sin embargo, la operación en sentido contrario se vuelve intratable por costes en computación y en tiempo, de forma que ahora para resolver este otro problema el tiempo será no polinómico (NP), lo que podríamos nuevamente asociar a un comportamiento exponencial del tiempo de respuesta.

Por este motivo, estas funciones también se conocen como funciones de un solo sentido.

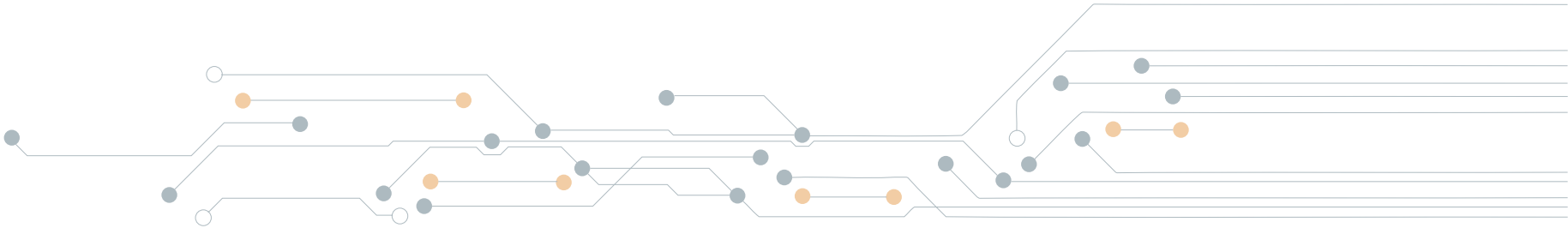
Estos problemas se usan en criptografía desde el invento de la criptografía de clave pública o asimétrica en 1976. Si los números en cuestión son relativamente grandes, sobre los mil bits, la solución al problema no polinómico se vuelve intratable, es decir no es viable con el cómputo actual.

Problema de la factorización entera PFE

El problema de la factorización entera (PFE) tiene relación con dos o más números primos y su producto. Para dos primos p y q el PFE indica lo siguiente:

- Multiplicar dichos primos p y q paras encontrar su producto (n = p x q), será una operación muy sencilla y rápida, de tipo polinómico (P). Es decir, si aumentamos el tamaño de los datos de entrada, el tiempo de cómputo de dicha multiplicación aumenta de forma directamente proporcional o lineal.

- Sin embargo, la operación inversa, que consiste en encontrar esos dos primos p y q si sólo conocemos el producto n entre ellos, se convierte en un cálculo de muy difícil solución, es decir no polinómico (NP), de forma que, si aumentamos ahora el tamaño de ese dato de entrada, el tiempo de cómputo para encontrar los dos primos aumentará de forma no lineal o exponencial.

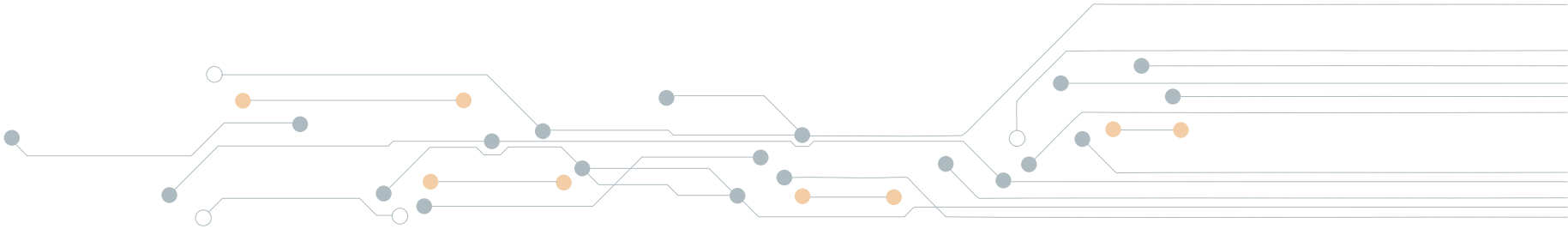


Problema del logaritmo discreto PLD

El Problema del logaritmo discreto (PLD) dice lo siguiente:

- Encontrar el resultado de la expresión mod p = , con p primo y un generador de ese primo, si se conocen p, y , es sencilla y rápida, incluso para números grandes. Una operación de tipo polinómica (P) o de comportamiento lineal.

- Sin embargo, conociendo , y n, encontrar ahora el valor del exponente = log mod n, se convierte en un problema de muy difícil solución para números grandes, una operación de tipo no polinómica (NP) o de comportamiento exponencial. Si bien la operación de calcular el logaritmo de en base es muy simple, el hecho de aplicar después la reducción módulo p al resultado, aporta una gran complejidad al problema.



Telefónica EDUCACIÓN DIGITAL