

CRIPTOGRAFÍA PARA INGENIER@S

Class4crypt

© Jorgeramió 2022

Aula virtual de
criptografía
aplicada

Diapositivas
utilizadas en las
clases grabadas
de Class4crypt

Módulo 2 Matemáticas discretas en la criptografía

Dr. Jorge Ramió Aguirre © 2022



Attribution-NonCommercial-
NoDerivatives 4.0 International
(CC BY-NC-ND 4.0)

Class4crypt

Tu aula virtual de criptografía aplicada



<https://www.youtube.com/user/jorgeramio>

Dr. Jorge Ramío Aguirre

*El ingenio es intrínseco al ser humano,
solo hay que darle una oportunidad
para que se manifieste.*

<https://www.criptored.es/cvJorge/index.html>

- ➔ Módulo 1. Principios básicos de la seguridad
- ➔ Módulo 2. Matemáticas discretas en la criptografía
- Módulo 3. Complejidad algorítmica en la criptografía
- Módulo 4. Teoría de la información en la criptografía
- Módulo 5. Fundamentos de la criptografía
- Módulo 6. Algoritmos de criptografía clásica
- Módulo 7. Funciones hash
- Módulo 8. Criptografía simétrica en bloque
- Módulo 9. Criptografía simétrica en flujo
- Módulo 10. Criptografía asimétrica

Class4crypt

Módulo 2. Matemáticas discretas en la criptografía

2.1. Aritmética modular, conjunto de restos y función de Euler

2.2. El homomorfismo de los enteros en la criptografía

2.3. Inverso aditivo, inverso xor e inverso multiplicativo

2.4. Cálculo de inversos con el algoritmo extendido de Euclides

2.5. Algoritmo de exponenciación modular rápida

2.6. Raíces primitivas en un primo p

Lista de reproducción del módulo 2 en el canal Class4crypt

<https://www.youtube.com/playlist?list=PLq6etZPDh0kvX0GdYngOtPcGYUhZ3XDKl>

Class4crypt c4c2.1

Módulo 2. Matemáticas discretas en la criptografía

Lección 2.1. Aritmética modular, conjunto de restos y función de Euler

2.1.1. Cuerpos, grupos, anillos, campos y módulo

2.1.2. Aritmética y operaciones modulares

2.1.3. Importancia de los primos y magnitudes

2.1.4. Conjunto completo y reducido de restos

2.1.5. Función de Euler $\phi(n)$

Class4crypt c4c2.1 Aritmética modular, conjuntos de restos y función de Euler

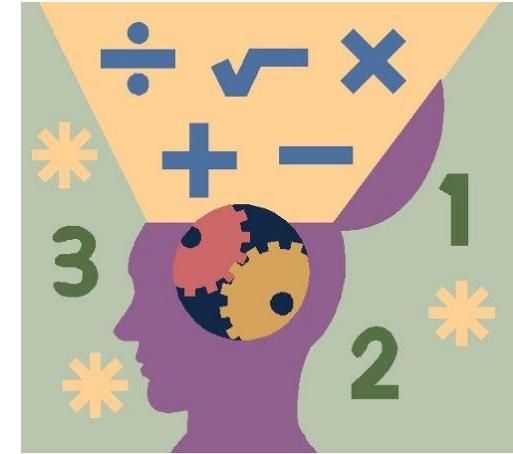
<https://www.youtube.com/watch?v=ExV55dvMbmI>

Cuerpos, grupos, anillos, campos... módulo

- Son conceptos con *gran calado* matemático pero no tan *fundamentales* para un ingeniero que se dedique a usar o implantar criptografía
- Por ello, en general llamaremos módulo al espacio de números enteros dentro del cual se realiza una operación de cifra, si bien hay diferencias importantes desde el punto de vista matemático de lo que es un cuerpo, un grupo, un anillo y un campo
- Para quien desee profundizar en este tema, se recomiendan estos dos documentos:
- “Cuerpos finitos”, Llorenç Huguet Rotger, Josep Rifà Coma, Juan Gabriel Tena Ayuso, Universitat Oberta de Catalunya (ver enlace en Lectura extra recomendada)
- “Grupos, anillos y cuerpos”, Manuel Palacios, Departamento de Matemática Aplicada, Centro Politécnico Superior, Universidad de Zaragoza (ver enlace en Lectura extra recomendada)

Aritmética modular

- Números enteros: ... -3, -2, -1, 0, 1, 2, 3...
- Tienen una gran aplicación en la criptografía
- Operaciones: reducción a módulo, suma y resta, producto, potencia o exponenciación, xor y la multiplicación por el inverso, puesto que “la división” no estará aquí permitida
- Conceptos de congruencia (notación correcta)



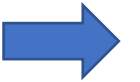
$$\begin{aligned}a - b &= k * n \\ a &\equiv_n b \\ a &\equiv b \bmod n\end{aligned}$$

Pregunta: ¿ $19 \equiv 4 \bmod 5$?
¿Es 19 congruente con 4 módulo 5?
Sí, porque $19 - 4 = 15 = k * 5$ (con $k = 3$)

Usaremos esta otra notación:

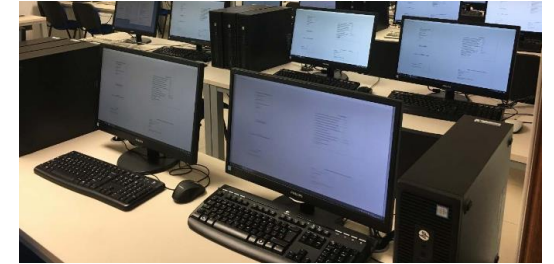
$$19 \bmod 5 = 4$$

Operaciones modulares

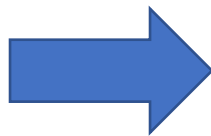


- Resto modular
 - $17 \bmod 20 = 17$; $45 \bmod 30 = 15$; $16.807 \bmod 7 = 0$; $-37 \bmod 27 = 17$
- Suma y resta modular
 - $45 + 33 \bmod 60 = 78 \bmod 60 = 18$; $58 - 75 \bmod 50 = -17 \bmod 50 = 33$
- Multiplicación modular
 - $127 \times 39 \bmod 1.001 = 4.953 \bmod 1.001 = 949$
- Potencia (exponenciación) modular
 - $98.717^5 \bmod 1.510 = 9.374.751.048.109.369.618.733.357 \bmod 1.510 = 377$
 - ¿Qué sucede ahora si deseamos calcular $5^{98.717}$? Es un número de 69.001 dígitos...
 - En criptografía (números muy grandes) no podrá hacerse x^y y “luego” reducir módulo n

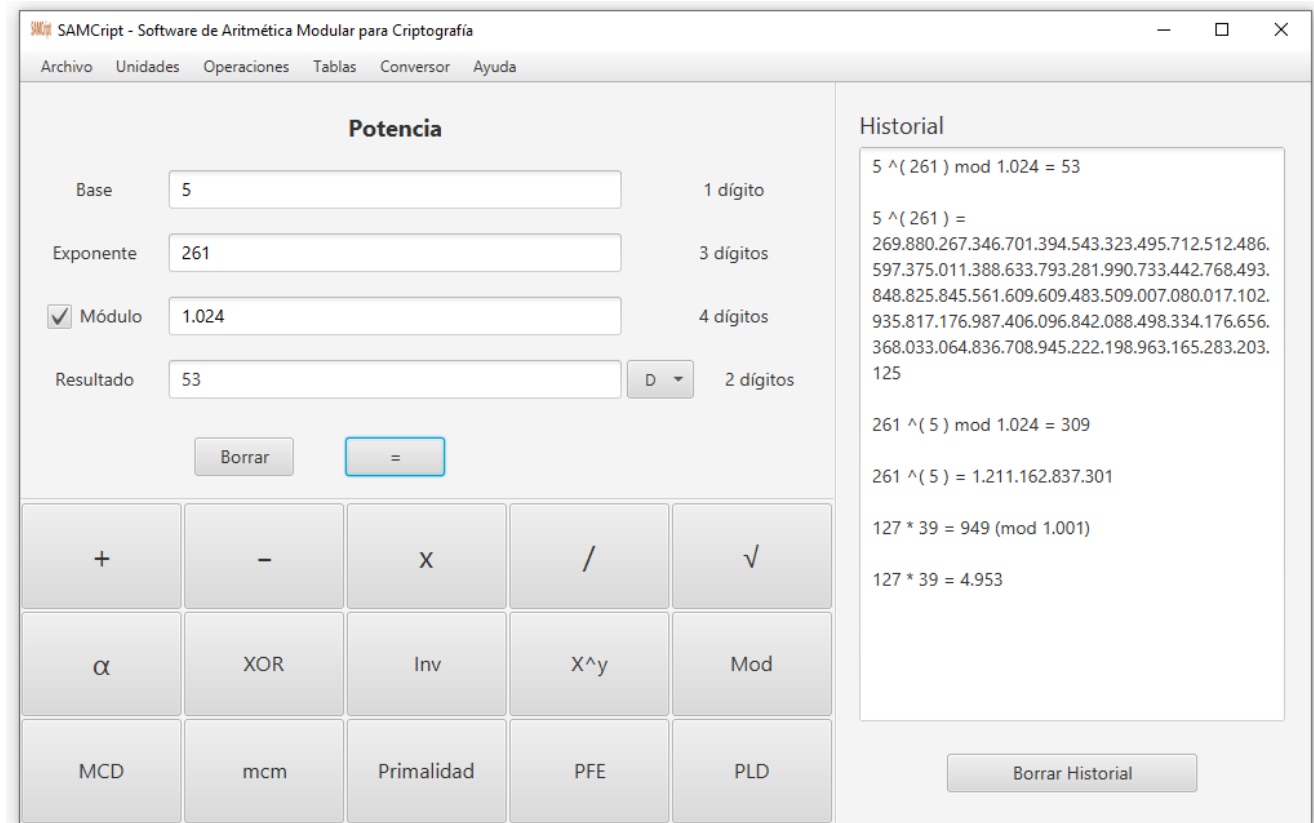
Laboratorio de prácticas



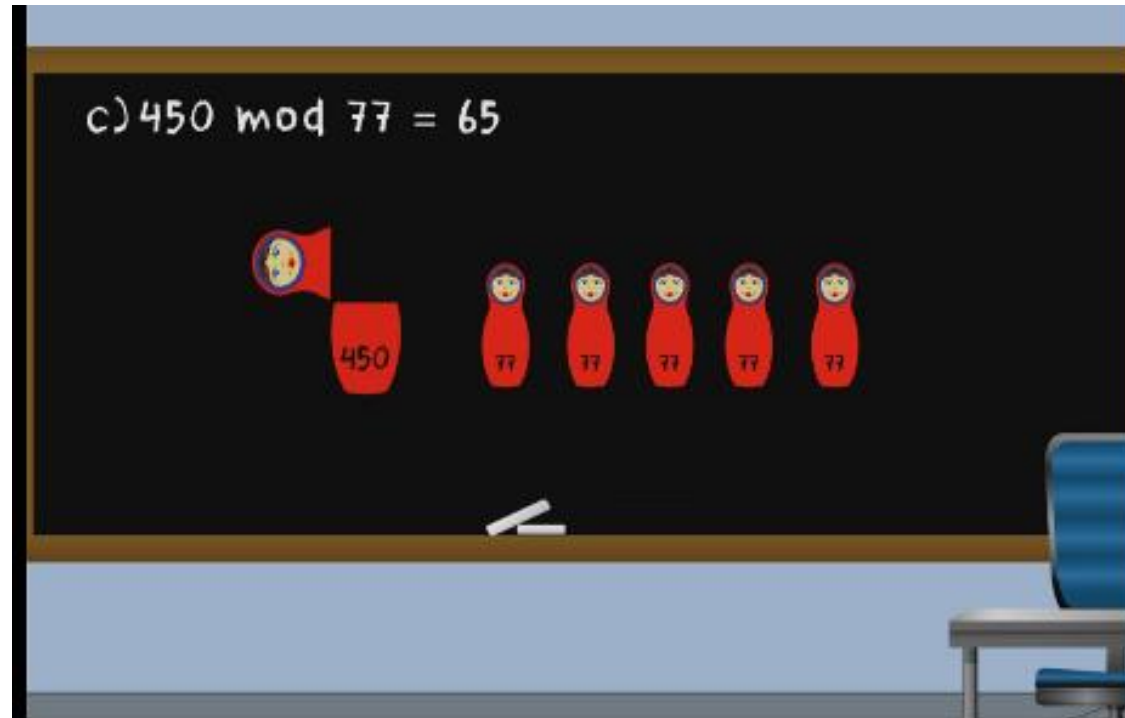
- Ejercicios prácticos de reducción a módulo, suma, resta, multiplicación y exponenciación modular,
- Manos a la obra...



Descarga SAMCrypt (ejecutable Java)
https://www.criptored.es/software/sw_m001t.htm



Más información en píldoras Thoth



<https://www.youtube.com/watch?v=y98MAZM2sqY>

Hay una errata obvia en 1:33 ($100 \bmod 77 = 23$ y no 33)

Números primos y números compuestos

- Las operaciones modulares en criptografía se realizarán dentro de un módulo de cifra, cuyo número puede ser primo o compuesto
- Un número es primo si tiene solo dos divisores, su mismo valor y 1
- Así por ejemplo, el 19 es un número primo porque sólo es divisible por 19 y por 1. Si dividimos 19 por los demás números menores que él (2, 3, 4,... 16, 17, 18), nunca obtendremos un número entero. En este caso se tiene:

$$19/2 = 9,50; 19/3 = 6,33; 19/4 = 4,75... 19/16 = 1,18; 19/17 = 1,11; 19/18 = 1,05$$

- Sin embargo, hablamos de un número compuesto si este está siempre representado por una multiplicación de dos o más números primos. Por ejemplo:

273 es compuesto porque es igual a $3 \cdot 7 \cdot 13$

2.200 es compuesto porque es igual a $2^3 \cdot 5^2 \cdot 11 = 8 \cdot 25 \cdot 11$

- Los números primos tendrán una gran importancia en la criptografía



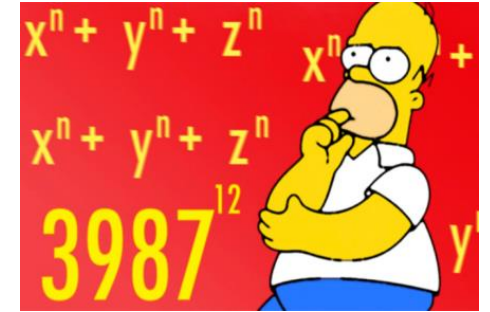
¿Cuántos primos hay ahí?



- Por el teorema de los números primos, se tiene que la probabilidad de encontrar números primos a medida que éstos se hacen más grandes es menor.
- En el intervalo $[2, x]$ habrá $\pi(x) \approx (x/\ln x)$ números primos
- Primos entre 2 y $2^5 = 32$ $x/\ln x = 32/3,46 = 9,25$ Probabilidad x sea primo: 28,90 %
- Primos entre 2 y $2^6 = 64$ $x/\ln x = 64/4,16 = 15,38$ Probabilidad x sea primo: 24,00 %
- Primos entre 2 y $2^7 = 128$ $x/\ln x = 128/4,85 = 26,40$ Probabilidad x sea primo: 20,62 %
- Primos entre 2 y $2^8 = 256$ $x/\ln x = 256/5,54 = 46,21$ Probabilidad x sea primo: 18,05 %
- Primos entre 2 y $2^9 = 512$ $x/\ln x = 512/6,23 = 82,18$ Probabilidad x sea primo: 16,05 %
- Primos entre 2 y $2^{10} = 1.024$ $x/\ln x = 1.024/6,93 = 147,76$ Probabilidad x sea primo: 14,43 %
- Primos entre 2 y $2^{11} = 2.048$ $x/\ln x = 2.048/7,62 = 268,77$ Probabilidad x sea primo: 13,12 %
- Primos entre 2 y $2^{12} = 4.096$ $x/\ln x = 4.096/8,32 = 492,31$ Probabilidad x sea primo: 12,02 %
- Etc.



¿Y cuántos primos de 1.024 bits?



¿Será primo?

- “Casi, casi” Homer: $3.987 + 2 = 3.989$ sí es primo
- Y el resultado de $3.987^{12} - 2 = p * q$ (144 bits)... casualidad
- $p * q = 103.515.353.519.882.399.087 * 155.865.521.984.159.680.926.017$
- $16.134.474.609.751.291.283.496.491.970.515.151.715.346.479$
- Los primos seguros (*safe primes*) serán interesantes en ciertos algoritmos de cifra como RSA. Un número primo p se dice que es seguro si $p = 2 * p' + 1$ (siendo p' también primo). Así, si $p' = 11$, luego $p = 2 * 11 + 1 = 23$, que es primo y por tanto, además, primo seguro

¿Cuántos primos hay con 1.024 bits significativos?

$$\pi(2^{1024}) - \pi(2^{1023}) \approx \frac{2^{1024}}{\ln 2^{1024}} - \frac{2^{1023}}{\ln 2^{1023}} \approx 1.26 \cdot 10^{305}$$

Referencia: https://en.wikipedia.org/wiki/Category:Classes_of_prime_numbers

↓ ¿?

¿Qué significa 10^{305} ?

- Criptografía y Seguridad en Computadores, Manuel Lucena, versión 5-0.1.4, nov 2019
- Capítulo 3, Números grandes, página 30
<http://criptografiayseguridad.blogspot.com/p/criptografia-y-seguridad-en.html>
- En criptografía moderna, nos preguntaremos qué esfuerzo de cómputo habría que hacer para romper una clave de 256 bits... 2^{256} intentos
- Ya se verá en próximas clases que un ataque de este tipo es materialmente imposible hacerlo

Valor	Número
Probabilidad de ser fulminado por un rayo (por día)	1 entre 9.000.000.000 (2^{33})
Probabilidad de ganar la Lotería Primitiva Española	1 entre 13.983.816 (2^{23})
Probabilidad de ganar la Primitiva y caer fulminado por un rayo el mismo día	1 entre 2^{56}
Tiempo hasta la próxima glaciación	14.000 (2^{14}) años
Tiempo hasta que el Sol se extinga	10^9 (2^{30}) años
Edad del Planeta Tierra	10^9 (2^{30}) años
Edad del Universo	10^{10} (2^{34}) años
Duración de un año	10^{13} (2^{44}) microsegundos
Número de átomos en el Planeta Tierra	10^{51} (2^{170})
Número de átomos en el Sol	10^{57} (2^{189})
Número de átomos en la Vía Láctea	10^{67} (2^{223})
Número de átomos en el Universo (excluyendo materia oscura)	10^{77} (2^{255})
Masa de la Tierra	$5,9 \times 10^{24}$ (2^{82}) Kg
Masa del Sol	2×10^{30} (2^{100}) Kg
Masa de la Vía Láctea	2×10^{42} (2^{140}) Kg
Masa estimada del Universo (excluyendo materia oscura)	10^{50} (2^{166}) Kg
Volumen de la Tierra	10^{21} (2^{69}) m ³
Volumen del Sol	10^{27} (2^{89}) m ³
Volumen estimado del Universo	10^{82} (2^{272}) m ³

Cuadro 1.1: Algunos números *grandes*

Conjunto completo de restos CCR

- Para cualquier entero positivo n , el conjunto completo de restos o CCR será:

$CCR = \{0, 1, 2, \dots, n-2, n-1\}$, es decir:

$$\forall a \in \mathbb{Z} \quad \exists ! r_i \in CCR / a \equiv r_i \pmod{n}$$

$$CCR(11) = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$CCR(6) = \{0, 1, 2, 3, 4, 5\} = \{12, -5, 20, 9, 16, 35\}$$

- El segundo conjunto de restos $\{12, -5, 20, 9, 16, 35\}$ en $n = 6$ es equivalente (clase de equivalencia) al primer conjunto de restos $\{0, 1, 2, 3, 4, 5\}$ porque:

12 es equivalente a 0 porque $12 \bmod 6 = 0$

-5 es equivalente a 1 porque $-5 \bmod 6 = 1$

20 es equivalente a 2 porque $20 \bmod 6 = 2$, etc.

- Normalmente (es recomendable) se trabajará en la zona positiva: $0, 1, 2, \dots, n-3, n-2, n-1$

Conjunto reducido de restos CRR

- El conjunto reducido de restos de n , conocido como CRR, es el subconjunto $\{0, 1, \dots, n_i, \dots, n-1\}$ de restos, primos relativos con el grupo n . Como el cero no es una solución, no se considera. Entonces:

$\text{CRR} = \{1, \dots, n_i, \dots, n-1\}$. Además, se cumplirá que $\text{mcd}(n_i, n) = 1$

- Si n es primo, obviamente todos los restos serán primos relativos con él y el CCR contendrá a todos los números excepto el 0; es decir, $n-1$ números

$$\text{CRR}(5) = \{1, 2, 3, 4\} \quad \text{CRR}(11) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

- Si n es un número compuesto, el CRR contendrá a los restos de n que no tengan factores en común con los factores primos de dicho número

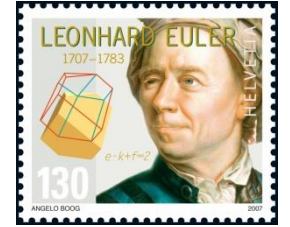
$\text{CRR}(8) = \{1, 3, 5, 7\}$. Como $8 = 2^3$, quitamos los números divisibles por 2 (2, 4, 6)

$\text{CRR}(15) = \{1, 2, 4, 7, 8, 11, 13, 14\}$, como $15 = 3 \cdot 5$, quitamos ahora 3, 6, 9, 12, 5 y 10

Importancia del CRR en criptografía

- El conocimiento del CRR permitirá aplicar un algoritmo para calcular el inverso multiplicativo de un resto a dentro de un grupo n a través de la función $\phi(n)$, denominada Función de Euler o Indicador de Euler
- El inverso multiplicativo (que se definirá y estudiará en una próxima transparencia) nos permitirá crear claves que sean inversas entre sí. Es decir, que la operación de cifra que se haga con una clave, se pueda deshacer (descifrar) con la otra clave, inversa de la primera
- La aritmética modular será muy importante en muchos sistemas de cifra simétricos. Una excepción muy particular es el algoritmo DES, al ser un caso especial de cifra no modular y que, además, no utiliza ninguna ecuación matemática al uso en su funcionamiento
- La Función de Euler $\phi(n)$ será vital en el algoritmo RSA, uno de los sistemas de cifra con clave pública o asimétrico más conocidos y populares, ya que los cálculos de inversos entre la clave pública y la clave privada se harán dentro de ese grupo

Función de Euler $\phi(n)$



- La Función de Euler $\phi(n)$ nos entregará la cantidad de restos del CRR
- Como podremos representar cualquier número n de las 4 formas que se indican a continuación, en cada una de ellas habrá una manera de calcular la Función de Euler $\phi(n)$

- 1) Si n es un número primo p ✓
- 2) Si n se representa como $n = p^k$ con p primo y k un entero
- 3) Si n es el producto $n = p * q$ con p y q primos ✓
- 4) Si n es un número cualquiera, con la forma genérica

$$n = p_1^{e1} * p_2^{e2} * \dots * p_t^{et} = \prod_{i=1}^t p_i^{ei}$$

Función de Euler $\phi(n)$ cuando $n = p$

- Si n es primo, $\phi(n)$ será igual al CCR menos el cero (0):

$$\phi(n) = n - 1$$

- Si n es primo, entonces $\text{CCR} = \text{CCR} - 1$ ya que todos los restos de n , excepto el cero, serán primos relativos o coprimos con p
- Ejemplo:

$\text{CCR}(7) = \{1, 2, 3, 4, 5, 6\}$, seis elementos. Luego $\phi(7) = n-1 = 7-1 = 6$

$$\phi(11) = 11-1 = 10; \quad \phi(23) = 23-1 = 22; \quad \phi(1.999) = 1.999-1 = 1.998$$

- Esta expresión se usará en los sistemas de cifra y firma de ElGamal, así como en la firma DSA (*Digital Signature Algorithm*) y DSS (*Digital Signature Standard*), en donde el módulo de cifra o de firma será un primo muy grande, superior a dos mil bits

Función de Euler $\phi(n)$ cuando $n = p*q$

- Si $n = p*q$ (con p y q primos), entonces:

$$\phi(n) = \phi(p*q) = \phi(p)*\phi(q) = (p-1)(q-1)$$

- De los $p*q$ elementos del CCR, quitaremos, todos los múltiplos de $p = 1*p, 2*p, 3*p, 4*p, \dots$ en general $(q-1)*p$, y todos los múltiplos de $q = 1*q, 2*q, 3*q, 4*q, \dots$ en general $(p-1)*q$, además del cero

$$\phi(p*q) = p*q - [(q-1) + (p-1) + 1] = p*q - q - p + 1 = (p-1)(q-1)$$

- $\text{CCR}(15) = \{1, 2, 4, 7, 8, 11, 13, 14\}$, ocho elementos
- En este caso $\phi(15) = \phi(3*5) = (3-1)(5-1) = 2*4 = 8$
- $\phi(306.997) = \phi(433*709) = (433-1)(709-1) = 432*708 = 305.856$

Uso de $\phi(n)$ cuando $n = p * q$

- Esta expresión se usará en el sistema de cifra RSA, y será una de las operaciones más utilizadas en la criptografía actual
- Es la base del sistema RSA, que durante muchos años ha sido un estándar. De hecho, continúa siéndolo, si bien en los últimos años ha cedido protagonismo en beneficio de los sistemas de cifra mediante curvas elípticas
- Actualmente, los primos p y q en RSA son números de 1.024 bits, por lo que el producto $n = p * q$ (el grupo de cifra y de firma) será de 2.048 bits y $\phi(n) = (p - 1)(q - 1)$ también será un número de 2.048 bits, ligeramente inferior a n
- Un uso típico: comunicaciones TLS con certificados digitales X.509, para comprobar la firma de una Autoridad de Certificación sobre un servidor web

Conclusiones de la lección 2.1

- La aritmética modular, o teoría de números, será muy importante en la criptografía
- Las operaciones típicas serán la reducción a módulo, la suma, la resta, la multiplicación y, muy especialmente, la exponenciación modular que se usará en la criptografía de clave pública o asimétrica
- Los números primos jugarán un papel muy importante en la criptografía
- El conjunto completo de restos y el conjunto reducido de restos tendrán implicaciones directas en el cálculo de las claves y en la cifra
- El indicador Euler o función de Euler, tendrá un papel especial en la cifra asimétrica RSA

Lectura recomendada (1/2)

- MOOC Crypt4you: Introducción a la seguridad informática y criptografía clásica. Tema III: Fundamentos básicos de matemáticas discretas, Jorge Ramió, 2016
 - <https://www.criptored.es/crypt4you/temas/criptografiaclasica/leccion5.html>
- Cuerpos finitos. Llorenç Huguet Rotger, Josep Rifà Coma, Juan Gabriel Tena Ayuso, Universitat Oberta de Catalunya (ver Lectura recomendada)
 - [https://www.exabyteinformatica.com/uoc/Informatica/Criptografia_avanzada/Criptografia_avanzada_\(Modulo_1\).pdf](https://www.exabyteinformatica.com/uoc/Informatica/Criptografia_avanzada/Criptografia_avanzada_(Modulo_1).pdf)
- Grupos, anillos y cuerpos. Manuel Palacios, Departamento de Matemática Aplicada, Centro Politécnico Superior, Universidad de Zaragoza
 - http://pcmap.unizar.es/~mpala/A_L_lecci/3grupos.pdf

Lectura recomendada (2/2)

- Por qué la segunda ley de la termodinámica pone límites a un ataque por fuerza bruta a una clave de 256 bits
 - <https://www.microsiervos.com/archivo/seguridad/segunda-ley-termodinamica-fuerza-bruta-256-bits.html>
- Guion píldora formativa Thoth nº 11, ¿Cifrando dentro de un cuerpo?, Jorge Ramió, 2014
 - <https://www.criptored.es/thoth/material/texto/pildora011.pdf>

Class4crypt c4c2.2

Módulo 2. Matemáticas discretas en la criptografía

Lección 2.2. El homomorfismo de los enteros en la criptografía

2.2.1. Recordando la aritmética modular

2.2.2. Homomorfismo de los enteros en operaciones modulares

2.2.3. Reducción por cuadrados en operaciones de exponenciación modular

2.2.4. Importancia del homomorfismo de los enteros en la cifra moderna

2.2.5. Calculadoras a usar para las operaciones de cifra con números grandes

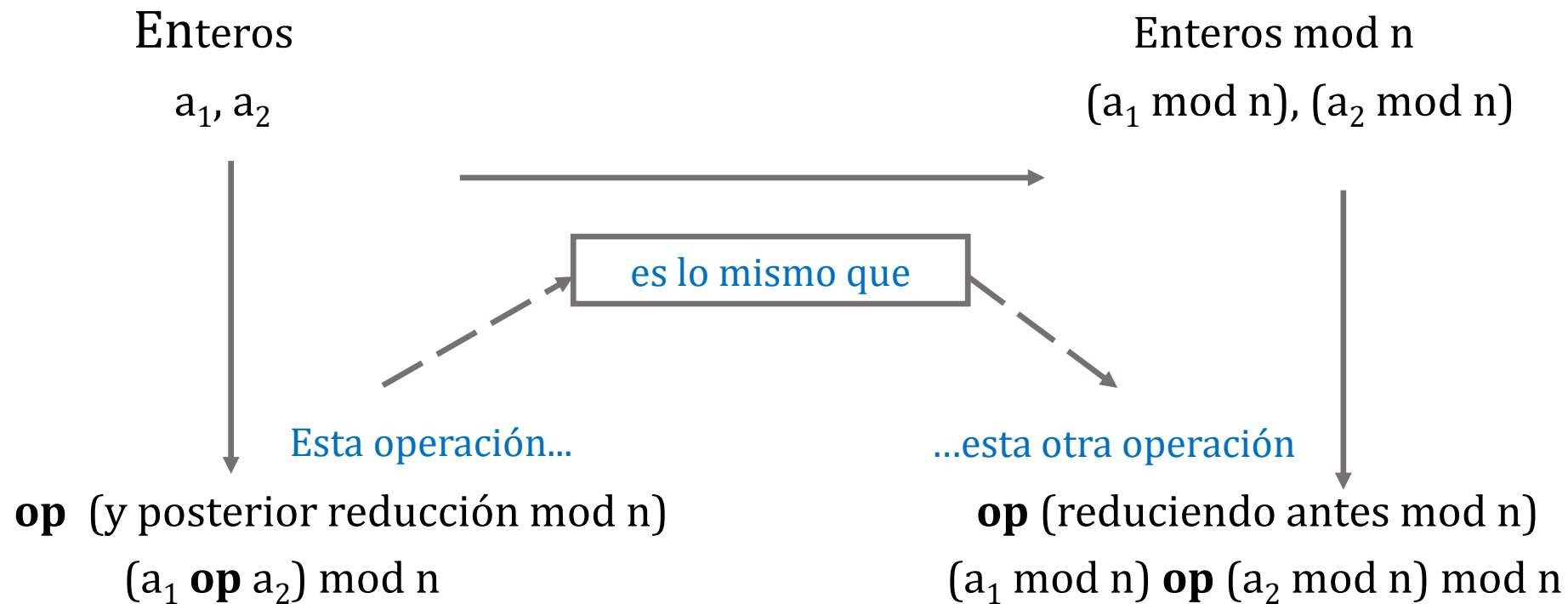
Class4crypt c4c2.2 El homomorfismo de los enteros en la criptografía
<https://www.youtube.com/watch?v=MNcXmz1qsA>

Recordando la aritmética modular


- En criptografía, las operaciones de cifra se hacen dentro de un módulo
- Sólo se cifran restos de ese módulo y los resultados de esa operación modular de cifra serán también restos de dicho módulo
- En la cifra clásica el módulo, grupo o cuerpo de cifra, tenía el mismo tamaño que el alfabeto
- **Operaciones:** $c_i = a * m_i + b \bmod 27$ (Afín), $c_i = m_i + k_i \bmod 27$ (Vigenère) ...
- En la cifra moderna esto no sucede, al tener como “alfabeto” el código ASCII extendido, bytes con sus $2^8 = 256$ caracteres, y siendo el módulo de cifra un número mucho mayor, por lo general miles de bits
- En criptografía asimétrica o de clave pública, ese módulo puede ser un primo p (cuerpo) o un número compuesto n (grupo), producto de dos primos
- **Operaciones:** $C_A = \alpha^a \bmod p$ (DH - Alicia), $C = K^{eR} \bmod n_R$ (RSA) ...

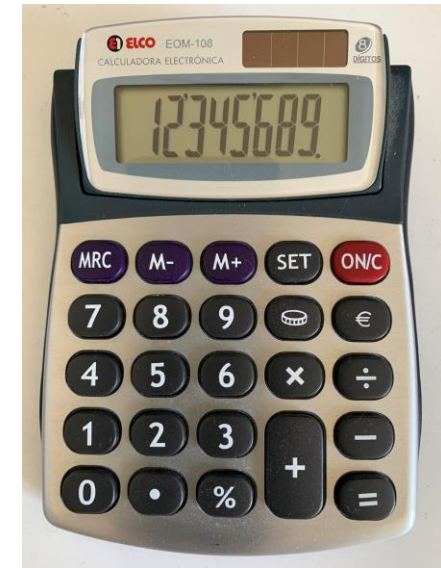
Homomorfismo de los enteros en módulo

En matemática el homomorfismo desde un objeto matemático a otro con la misma estructura algebraica, es una función que preserva las operaciones definidas en dichos objetos. Esto nos permitirá realizar operaciones modulares con números muy grandes en la criptografía actual



Reduciendo las operaciones modulares

- $315^{12} \bmod 401 = 954.391.681.757.526.337.812.744.140.625 \bmod 401 = 125$
- Al ser números muy pequeños podemos usar la calculadora de Windows
- Como $315^{12} = 315^{2*6}$, esta operación (op) también se podría resolver así:
- $315^2 * 315^2 * 315^2 * 315^2 * 315^2 * 315^2 \bmod 401$
- Como $315^2 \bmod 401 = 178$, tenemos:
- $178^2 * 178^2 * 178^2 \bmod 401 = 5 * 5 * 5 = 125$ 
- La reducción por cuadrados no será eficiente para números muy grandes
- Aquí usaremos el Algoritmo de Exponenciación Rápida (AER), que veremos en una próxima clase y que está basado en esta misma propiedad



Ejemplo práctico

¿Por qué es importante el homomorfismo?

- El homomorfismo de los enteros será muy importante en criptografía de clave pública, en donde muchas operaciones son del tipo $A^b \bmod n$, con valores de A y b que pueden ser incluso de miles de bits, números muy grandes
- Intentar resolver primero la potencia A^b y después reducir el resultado al módulo n , es imposible. Entre otras razones, porque no hay manera de almacenar ese número inmenso en ningún ordenador o sistema.
- Por ejemplo, $(256 \text{ bits})^{(17 \text{ bits})}$ ya entrega más de 16 millones de bits...
- Sin embargo, gracias al homomorfismo de los enteros, las operaciones pueden hacerse incluso con números del mismo orden de bits que el módulo n de cifra, como sería el caso del descifrado de una clave de sesión tras un intercambio de clave realizado con RSA
- Es decir, realizar una operación $(2.048 \text{ bits})^{(2.048 \text{ bits})} \bmod (2.048 \text{ bits})$

El homomorfismo en la cifra asimétrica

- En criptografía moderna asimétrica trabajamos con números muy grandes, porque la fortaleza de los algoritmos reside en un problema matemático de muy difícil solución cuando el módulo es grande, llamados problemas NP (nondeterministic polynomial time) o tiempo polinomial no determinista
- A pesar de ello, hay que asegurar que podremos realizar una operación del tipo $(256 \text{ bits})^{(17 \text{ bits})} \bmod (2.048 \text{ bits})$, intercambio de clave AES256 con RSA, o $(256 \text{ bits})^{(2.048 \text{ bits})} \bmod (2.048 \text{ bits})$, firma digital de un hash SHA-2 con RSA
- Pero no podremos calcular N^x y “después” reducir módulo n (imposible...)
 - $243^{19} = 2.120.895.147.045.314.119.491.609.587.512.844.743.630.072.107$ (46 dígitos, 151 bits)
 - Pero si intercambiamos esos números, aun siendo extremadamente pequeños, $a: 19^{243} = 54.591.500.508.179.050.937.318.088.200.925.014.403.198.664.275.313.933.277.242.918.098.453.270.394.419.502.761.682.134.122.372.615.632.042.338.880.435.800.873.587.880.321.946.719.404.884.689.789.912.724.211.616.943.306.657.280.907.354.665.390.070.892.406.881.184.847.540.695.653.985.526.095.057.496.074.324.352.268.540.317.465.180.142.907.170.564.291.497.461.564.034.165.111.403.756.258.878.363.860.020.756.571.659$ (311 dígitos = 1.033 bits)

Operaciones de potencia N^x y $N^x \bmod n$

- Con números grandes no podemos calcular N^x pero sí podremos calcular $N^x \bmod n$ usando el homomorfismo de los enteros. Por ejemplo, este intercambio de clave K de 256 bits del algoritmo AES mediante el algoritmo RSA de 2.048 bits (tiempo \ll 1 segundo)
- $C8FC0451D2D7E4E34B2319D7D043732C85EEEB4843D078A0ADB235E34F4C401E^{10001} \bmod$
C21F3BC0C9B72ECCCFB22F21EAA218F188C43D4E2ADC713D5307330B188AB9869E251AC20F832
F6A0FB5B23CB8DF3BAAF271D9D34E7EC202FAC640E2C3D0EBB605DFE5FC352736FF9CFE17D671
6F127EEAF3CAA05AD1763D1717E7DF88924D757155F3142844EB46DD13D797867AC23AD053227
B2EEA5C904498DA8680D028CE955ABB1391C15F583D0C8E70536428C629FCCF493537E04356DC8
CBA19F3916B29B0E03179A96F61BD8C44433B690F4B6D9BEBF15BD638CD0F22544FDA5A87946E
B3AF75CAB029F84B786C500CF95529DFDCE6E21E0A8820E151A9212963119A8E315E5F514FAA58
4DB8AC3099F568AC84975A70FD5E515A37966A1245A3E55F
- BFA103B8E9A64109674151FE61121B6312BA634EA835BB76CC44BC64A396C54AC0CF6C70C85387
8A39A69EA5A8E1CD5B9304B0B086BB0A2F17AD905BFA4B162627C7E3A75161D5B9EB2F096136C
B9F5A22E79E70135266B191CE4FABA9F7D5A1EA34801A66619DA0B87C9773F6AF07C23D5C89F9
9F5F248F918DC2D2D1B5441E509F648B8C9620DBFCAE4EB12C2C4D81A8B1F56F3C3721C2A327F
3EC7CAD3D10B46042F7586261AD67A60D19A4C309549B2135999F054A1F4ED9B3BB31D0A2E007
BA49D4EB12F166B2EAB595988DE5F9D59E404C448CF44A035DB12DBC43AF6BE665DC5B28CF86C
4111786359E7E2676D812A97C6ED26389C5A0DC188598E06F

Cálculo de $N^x \bmod n$ de forma inmediata

SAMCrypt - Software de Aritmética Modular para Criptografía

Archivo Unidades Operaciones Tablas Conversor Ayuda

Potencia

Base: C8FC0451D2D7E4E34B2319D7D043732C85EEEB48431 64 caracteres

Exponente: 10001 5 caracteres

☒ Módulo: c21f3bc0c9b72eccfb22f21eaa218f188c43d4e2adc71e 512 caracteres

Resultado: BFA103B8E9A64109674151FE61121B6312BA634EA83! 512 caracteres

Borrar =

+	-	x	/	√
α	XOR	Inv	X^y	Mod
MCD	mcm	Primalidad	PFE	PLD

Historial

70536428c629fccf493537e04356dc8cba19f391
6b29b0e03179a96f61bd8c44433b690f4b6d9be
bf15bd638cd0f22544fda5a87946eb3af75cab02
9f84b786c500cf95529dfdce6e21e0a8820e151a
9212963119a8e315e5f514faa584db8ac3099f56
8ac84975a70fd5e515a37966a1245a3e55f =
BFA103B8E9A64109674151FE61121B6312BA63
4EA835BB76CC44BC64A396C54AC0CF6C70C85
3878A39A69EA5A8E1CD5B9304B0B086BB0A2F
17AD905BFA4B162627C7E3A75161D5B9EB2F0
96136CB9F5A22E79E70135266B191CE4FABA9F
7D5A1EA34801A66619DA0B87C9773F6AF07C2
3D5C89F99F5F248F918DC2D2D1B5441E509F6
48B8C9620DBFCAE4EB12C2C4D81A8B1F56F3C
3721C2A327F3EC7CAD3D10B46042F7586261A
D67A60D19A4C309549B2135999F054A1F4ED9
B3BB31D0A2E007BA49D4EB12F166B2EAB5959
88DE5F9D59E404C448CF44A035DB12DBC43AF
6BE665DC5B28CF86C4111786359E7E2676D81
2A97C6ED26389C5A0DC188598E06F

Borrar Historial

Interesante:

Clave 256 bits (AES)
Clave pública F4 (RSA)
256 bits¹⁷ bits

0x
C8FC0451D2D7E4E34B2319D7D043
732C85EEEB4843D078A0ADB235E34
F4C401E¹⁰⁰⁰¹

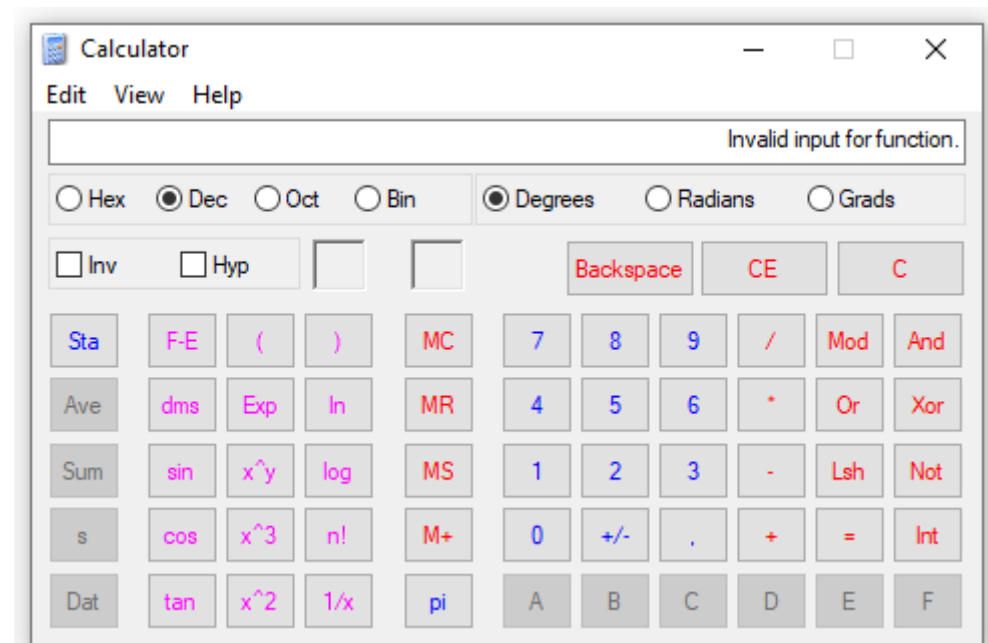
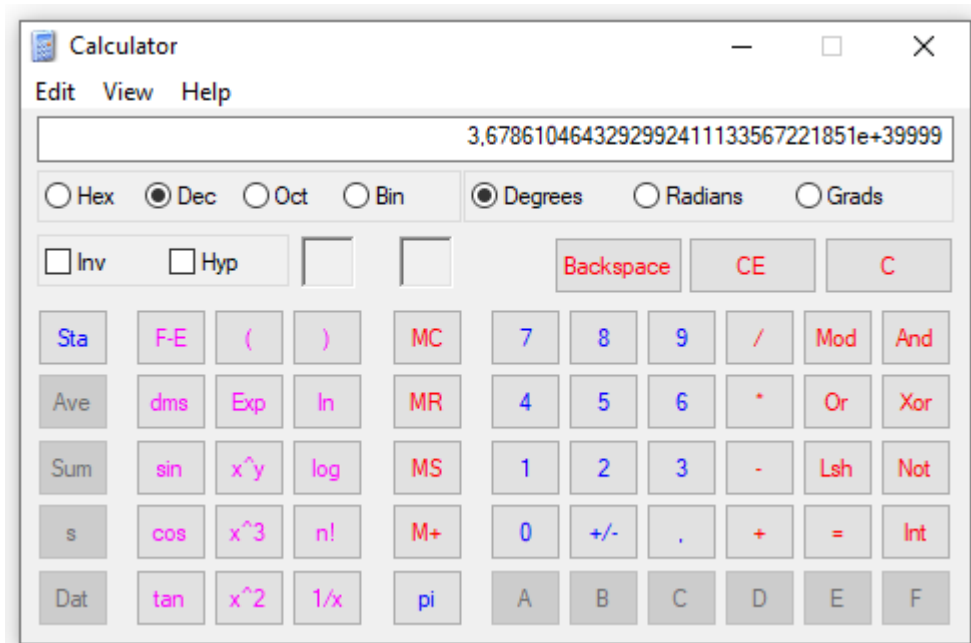
4.188.649 caracteres hexadecimales

16.754.596 bits (mucho mayor que
el módulo de cifra de 2.048 bits)

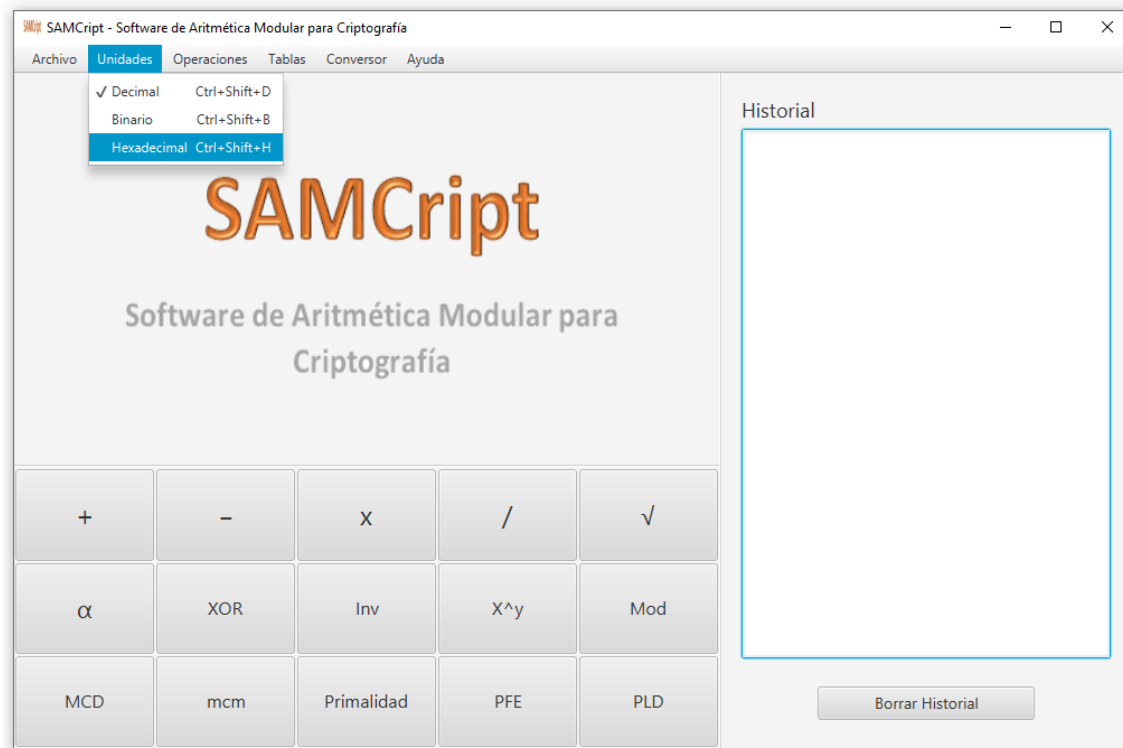
¿Qué calculadora usar, Windows?

- $9.999^{10.000} = 3,6786104643292992411133567221851e+39999$ (40 mil dígitos)
- Pero $9.999^{10.999}$ = entrada no válida
- > 5 dígitos^{5 dígitos} (por ejemplo $11.111^{11.111}$): entrada no válida

¿Solución? ➡



Software para operaciones modulares



SAMCrypt (recomendable)

https://www.criptored.es/software/sw_m001t.htm

Equation	Description	Example	Equation to be used in this tool
$a + b$	Add a and b.	$3 + 2 = 5$	<code>add(a,b)</code>
$a + n$	Add a and n	$3 + 2 = 5$	<code>addInt(a,n)</code>
$a * b$	Multiply a and b	$3 * 2 = 6$	<code>mult(a,b)</code>
$a - b$	Subtract a and b	$3 - 2 = 1$	<code>sub(a,b)</code>
$\gcd(a,b)$	Calculate greatest common divisor of a and b.	$\gcd(36,24) = 12$	<code>GCD(a,b)</code>
$a ^ b$	Calculate $a ^ b$	$2 ^ 3 = 8$	<code>bigPow(a,b)</code>
$a \% b$	Calculate a modulo b	$11 \% 7 = 4$	<code>mod(a,b)</code>
$(a * b) \% c$	Multiply a and b. Calculate result modulo c	$(3 * 5) \% 9 = 6$	<code>multMod(a,b,c)</code>
$(a ^ b) \% c$	Calculate a^b . Calculate result modulo c	$(3 ^ 2) \% 6 = 3$	<code>powMod(a,b,c)</code>
$(a ^{-1}) \% b$	Calculate a^{-1} . Calculate result modulo b	$(2 ^{-1}) \% 3 = 2$	<code>inverseMod(a,b)</code>
Examples			
$(a + b) * (c + d)$	-	$(1 + 2) * (3 + 4) = 21$	<code>mult(add(a,b), add(c,d))</code>
$(a - b) * (c - d)$	-	$(2 - 1) * (4 - 2) = 2$	<code>mult(sub(a,b), sub(c,d))</code>
$(a - 1) * (b - 1)$	-	$(5 - 1) * (7 - 1) = 24$	<code>mult(addInt(a,-1), addInt(b,-1))</code>
$a + b + c + d + e$	-	$1 + 2 + 3 + 4 + 5 = 15$	<code>add(add(add(add(a,b),c),d),e)</code>

Note:

a, b, c, d or e are big positive integer numbers.

n is a small integer number

Mobilefish (no cómodo)

https://www.mobilefish.com/services/big_number_equation/big_number_equation.php

Cuidado con el software en Internet

[illegible]

$$\text{inv}(31, 200000000000000000000) = 83.870.967.741.935.483.871$$

[illegible]

Sin palabras...

Conclusiones de la lección 2.2

- El homomorfismo de los enteros permite realizar operaciones modulares con números muy grandes del tipo $N^x \bmod n$, típicas en criptografía asimétrica
- El método por reducción de cuadrados es válido para números grandes porque optimiza el cómputo, pero no para números muy grandes
- La extensión de este método se conoce como Algoritmo de Exponenciación Rápida AER, que veremos en una próxima clase
- La calculadora de Windows permite hacer operaciones de exponenciación sólo hasta unos 44 mil dígitos, demasiado pequeño para la criptografía actual
- Para cálculos con números muy grandes, se recomienda utilizar el software SAMCrypt en vez de Mobilefish (incómodo y consume muchos recursos web)
- Cuidado con páginas web de matemáticas en Internet

Lectura recomendada

- Congruencias en \mathbb{Z} módulo m , La exponenciación en aritmética modular
 - http://www.dma.fi.upm.es/recursos/aplicaciones/matematica_discreta/web/aritmetica_modular/congruencias.html
- Exponenciación modular rápida, Khan Academy
 - <https://es.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/fast-modular-exponentiation>
- Modular exponentiation, Khan Academy
 - <https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-exponentiation>
- Modular exponentiation, Wikipedia
 - https://en.wikipedia.org/wiki/Modular_exponentiation

Class4crypt c4c2.3

Módulo 2. Matemáticas discretas en la criptografía

Lección 2.3. Inverso aditivo, inverso xor e inverso multiplicativo

2.3.1. Importancia de los inversos en la criptografía

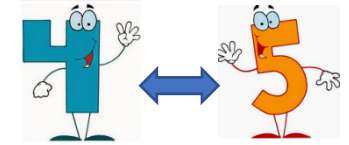
2.3.2. Los inversos aditivos

2.3.3. Los inversos xor

2.3.4. Los inversos multiplicativos

Class4crypt c4c2.3 Inverso aditivo, inverso xor e inverso multiplicativo
<https://www.youtube.com/watch?v=7jaoEg1g10c>


Criptografía e inversos



- En criptografía deberá estar permitido invertir una operación para recuperar el texto en claro de un texto cifrado (criptograma). Es lo que se conoce como operación de descifrado
- Aunque la cifra sea una función, en lenguaje coloquial la operación de cifrado podría interpretarse como una “multiplicación” y la operación de descifrado como una “división”, si bien hablaremos en este caso de una multiplicación por el inverso. La analogía anterior sólo será válida en los números enteros \mathbb{Z} con inverso
- La división está prohibida en matemática discreta. Por ejemplo, si se cifra en mod 27 (mayúsculas del español) multiplicando los códigos de las letras (A=0, B=1, ... Y=25, Z=26) por 4, para descifrar no se puede dividir por 4 (ej. $17/4 = 4,25$ no es entero)
- Habrá que multiplicar por $\text{inv}(4, 27) = 7$, que en el fondo lo mismo
- Por tanto, si en una operación de cifra la función es multiplicar por el valor a dentro de un módulo n , deberemos encontrar el inverso $a^{-1} \bmod n$ para descifrar

¿Por qué usamos inversos en criptografía?

- En la criptografía clásica y en la moderna debemos ser capaces de descifrar lo que se ha cifrado previamente, es decir, deshacer la operación de cifra
- Cifra clásica (desplazamiento puro y decimación pura)
- Cifrado: $c_i = m_i + 6 \bmod 27$ Descifrado: $m_i = c_i + 21 \bmod 27$ ✓
- Cifrado: $c_i = 7 * m_i \bmod 27$ Descifrado: $m_i = c_i * 4 \bmod 27$ ✓
- Cifra moderna asimétrica RSA
- Cifrado: $C = N^7 \bmod 77$ Descifrado: $N = C^{43} \bmod 77$ ✓ USO DE INVERSOS

- ¿Si el módulo es primo, el exponente indica el tipo de clave? NO
- Cifrado: $C = N^7 \bmod 41$ Descifrado: $N = C^{23} \bmod 41$ (sí funciona...)
- **PERO** esto no siempre es válido $C = N^4 \bmod 41$ (ya no se descifra) 

$N^7 \bmod 41$ se descifra como $C^{23} \bmod 41 \dots$

$$C = N^x \bmod p = N^7 \bmod 41$$

$$N = C^y \bmod p = C^{23} \bmod 41$$

Caso especial de cifrado y de descifrado exponencial modular dentro de un primo, en donde se cumple que cada resto N del módulo p se cifra diferente ($N^x \bmod p = C$) y que el descifrado se realiza con un único valor como exponente ($C^y \bmod p = N$)

Pero...

$0^7 \bmod 41 = 0$	$1^7 \bmod 41 = 1$	$2^7 \bmod 41 = 5$	$3^7 \bmod 41 = 14$	$4^7 \bmod 41 = 25$
$5^7 \bmod 41 = 20$	$6^7 \bmod 41 = 29$	$7^7 \bmod 41 = 17$	$8^7 \bmod 41 = 2$	$9^7 \bmod 41 = 32$
$10^7 \bmod 41 = 18$	$11^7 \bmod 41 = 35$	$12^7 \bmod 41 = 22$	$13^7 \bmod 41 = 26$	$14^7 \bmod 41 = 3$
$15^7 \bmod 41 = 34$	$16^7 \bmod 41 = 10$	$17^7 \bmod 41 = 13$	$18^7 \bmod 41 = 37$	$19^7 \bmod 41 = 30$
$20^7 \bmod 41 = 8$	$21^7 \bmod 41 = 33$	$22^7 \bmod 41 = 11$	$23^7 \bmod 41 = 4$	$24^7 \bmod 41 = 28$
$25^7 \bmod 41 = 31$	$26^7 \bmod 41 = 7$	$27^7 \bmod 41 = 38$	$28^7 \bmod 41 = 15$	$29^7 \bmod 41 = 19$
$30^7 \bmod 41 = 6$	$31^7 \bmod 41 = 23$	$32^7 \bmod 41 = 9$	$33^7 \bmod 41 = 39$	$34^7 \bmod 41 = 24$
$35^7 \bmod 41 = 12$	$36^7 \bmod 41 = 21$	$37^7 \bmod 41 = 16$	$38^7 \bmod 41 = 27$	$39^7 \bmod 41 = 36$
$40^7 \bmod 41 = 40$	Cifrado $N^x \bmod p$ con $p = 41$ y clave $x = 7$, para todos los restos de p			

$0^{23} \bmod 41 = 0$	$1^{23} \bmod 41 = 1$	$2^{23} \bmod 41 = 8$	$3^{23} \bmod 41 = 14$	$4^{23} \bmod 41 = 23$
$5^{23} \bmod 41 = 2$	$6^{23} \bmod 41 = 30$	$7^{23} \bmod 41 = 26$	$8^{23} \bmod 41 = 20$	$9^{23} \bmod 41 = 32$
$10^{23} \bmod 41 = 16$	$11^{23} \bmod 41 = 22$	$12^{23} \bmod 41 = 35$	$13^{23} \bmod 41 = 17$	$14^{23} \bmod 41 = 3$
$15^{23} \bmod 41 = 28$	$16^{23} \bmod 41 = 37$	$17^{23} \bmod 41 = 7$	$18^{23} \bmod 41 = 10$	$19^{23} \bmod 41 = 29$
$20^{23} \bmod 41 = 5$	$21^{23} \bmod 41 = 36$	$22^{23} \bmod 41 = 12$	$23^{23} \bmod 41 = 31$	$24^{23} \bmod 41 = 34$
$25^{23} \bmod 41 = 4$	$26^{23} \bmod 41 = 13$	$27^{23} \bmod 41 = 38$	$28^{23} \bmod 41 = 24$	$29^{23} \bmod 41 = 6$
$30^{23} \bmod 41 = 19$	$31^{23} \bmod 41 = 25$	$32^{23} \bmod 41 = 9$	$33^{23} \bmod 41 = 21$	$34^{23} \bmod 41 = 15$
$35^{23} \bmod 41 = 11$	$36^{23} \bmod 41 = 39$	$37^{23} \bmod 41 = 18$	$38^{23} \bmod 41 = 27$	$39^{23} \bmod 41 = 33$
$40^{23} \bmod 41 = 40$	Se descifran todos los números cifrados usando como exponente el 23			

Pero con el exponente 4 no funciona...

$0^4 \bmod 41 = 0$	$1^4 \bmod 41 = 1$	$2^4 \bmod 41 = 16$	$3^4 \bmod 41 = 40$	$4^4 \bmod 41 = 10$
$5^4 \bmod 41 = 10$	$6^4 \bmod 41 = 25$	$7^4 \bmod 41 = 23$	$8^4 \bmod 41 = 37$	$9^4 \bmod 41 = 1$
$10^4 \bmod 41 = 37$	$11^4 \bmod 41 = 4$	$12^4 \bmod 41 = 31$	$13^4 \bmod 41 = 25$	$14^4 \bmod 41 = 40$
$15^4 \bmod 41 = 31$	$16^4 \bmod 41 = 18$	$17^4 \bmod 41 = 4$	$18^4 \bmod 41 = 16$	$19^4 \bmod 41 = 23$
$20^4 \bmod 41 = 18$	$21^4 \bmod 41 = 18$	$22^4 \bmod 41 = 23$	$23^4 \bmod 41 = 16$	$24^4 \bmod 41 = 4$
$25^4 \bmod 41 = 18$	$26^4 \bmod 41 = 31$	$27^4 \bmod 41 = 40$	$28^4 \bmod 41 = 25$	$29^4 \bmod 41 = 31$
$30^4 \bmod 41 = 4$	$31^4 \bmod 41 = 37$	$32^4 \bmod 41 = 1$	$33^4 \bmod 41 = 37$	$34^4 \bmod 41 = 23$
$35^4 \bmod 41 = 25$	$36^4 \bmod 41 = 10$	$37^4 \bmod 41 = 10$	$38^4 \bmod 41 = 40$	$39^4 \bmod 41 = 16$
$40^4 \bmod 41 = 1$				

$25^0 \bmod 41 = 1$	$25^1 \bmod 41 = 25$	$25^2 \bmod 41 = 10$	$25^3 \bmod 41 = 4$	$25^4 \bmod 41 = 18$
$25^5 \bmod 41 = 40$	$25^6 \bmod 41 = 16$	$25^7 \bmod 41 = 31$	$25^8 \bmod 41 = 37$	$25^9 \bmod 41 = 23$
$25^{10} \bmod 41 = 1$	$25^{11} \bmod 41 = 25$	$25^{12} \bmod 41 = 10$	$25^{13} \bmod 41 = 4$	$25^{14} \bmod 41 = 18$
$25^{15} \bmod 41 = 40$	$25^{16} \bmod 41 = 16$	$25^{17} \bmod 41 = 31$	$25^{18} \bmod 41 = 37$	$25^{19} \bmod 41 = 23$
$25^{20} \bmod 41 = 1$	$25^{21} \bmod 41 = 25$	$25^{22} \bmod 41 = 10$	$25^{23} \bmod 41 = 4$	$25^{24} \bmod 41 = 18$
$25^{25} \bmod 41 = 40$	$25^{26} \bmod 41 = 16$	$25^{27} \bmod 41 = 31$	$25^{28} \bmod 41 = 37$	$25^{29} \bmod 41 = 23$
$25^{30} \bmod 41 = 1$	$25^{31} \bmod 41 = 25$	$25^{32} \bmod 41 = 10$	$25^{33} \bmod 41 = 4$	$25^{34} \bmod 41 = 18$
$25^{35} \bmod 41 = 40$	$25^{36} \bmod 41 = 16$	$25^{37} \bmod 41 = 31$	$25^{38} \bmod 41 = 37$	$25^{39} \bmod 41 = 23$
$25^{40} \bmod 41 = 1$				

$$C = N^x \bmod p = N^4 \bmod 41$$

$$N = C^y \bmod p = C^{i?} \bmod 41$$

- Para $x = 4$ la cifra entrega valores repetidos, lo cual es inaceptable en criptografía
- Nunca se recupera el texto en claro. Por ejemplo, 6, 13, 28 y 35 se cifran como 25, pero al descifrar $C = 25$ con $y = 0$ hasta $y = 40$ no se obtiene N
- La solución a esto vendrá con el algoritmo de Elgamal

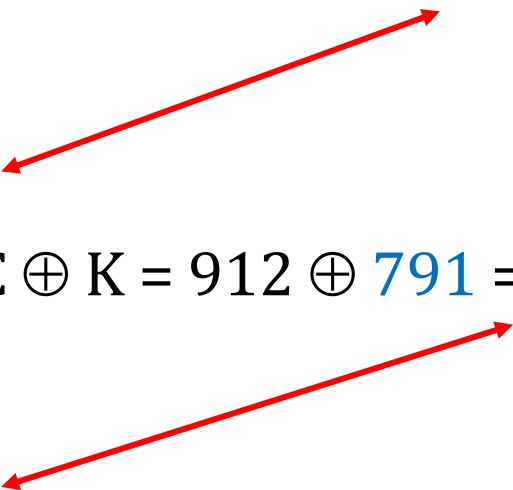
Inversos aditivos $x = \text{inv}_+(a, n)$

- El inverso aditivo de un resto a de n , dentro de ese módulo n , siempre existirá
- Será el complemento al módulo. Esto es, el valor x que sumado al resto a , entrega la identidad de la suma (0) dentro de ese módulo n
- Por ejemplo, en el módulo 31 el inverso aditivo de 12 será 19 porque $12 + 19 \bmod 31 = 0$
- Es decir, $\text{inv}_+(12, 31) = -12 \bmod 31 = (-12 + 31) \bmod 31 = 19 \bmod 31$
- Los valores -12 y 19 (entre otros) son clases de equivalencia en el módulo 31
- El algoritmo IDEA usa una operación suma modular $\bmod 65.536$ (2^{16}) con la clave Z_2 de 16 bits y la segunda palabra de 16 bits del bloque de entrada de 64 bits. En el descifrado, en esa zona se usará el inverso. Si $Z_2 = 39.057$, ¿cuál es el valor de ese inverso $-Z_2$? Solución: $-Z_2 = 65.536 - 39.057 = 26.479$

Inversos xor $\text{inv}_{\oplus}(x)$

- El inverso xor de un número x siempre existirá y es el mismo, ya que xor es una función involutiva: si se aplica dos veces, se vuelve al estado original
- Por ejemplo sea el secreto el número $N = 135$ a enviar mediante una cifra xor con la clave secreta compartida por emisor y receptor $K = 791$

- Cifrado: $C = N \oplus K = 135 \oplus 791 = 912$

$$\begin{array}{r} 0010000111 \\ \oplus 1100010111 \\ \hline 1110010000 \end{array}$$


- Descifrado: $N = C \oplus K = 912 \oplus 791 = 135$

$$\begin{array}{r} 1110010000 \\ \oplus 1100010111 \\ \hline 0010000111 \end{array}$$

El algoritmo IDEA usa una operación suma mod 2 con la clave Z_5 de 16 bits y una palabra de 16 bits calculada previamente. En esa zona de descifrado se usará el inverso. Si $Z_5 = 53.904$, ¿cuál es el valor de ese inverso xor? Solución: $\text{inv}_{\oplus} 53.904 = 53.904$

Más información en píldoras Thoth



<https://www.youtube.com/watch?v=1Jg-HVoA6-M>

Inversos multiplicativos $x = \text{inv}(a, n)$

- El inverso multiplicativo de un resto a de n , dentro de ese módulo n , no siempre existirá, será un caso especial
- El inverso multiplicativo del resto a en n , será el valor x que multiplicado por a y reducido mod n entrega la identidad de la multiplicación (1)
- Condición necesaria para que el inverso multiplicativo exista
 - $\text{MCD}(a, n) = 1$
- Si $n = p$ (primo), entonces el inverso multiplicativo siempre existirá
- Por ejemplo para $p = 31$ y para $n = 87 = 3 \cdot 29$
 - $\text{inv}(15, 31) = 29$ Ya que $29 \cdot 15 \bmod 31 = 435 \bmod 31 = 14 \cdot 31 + 1 \bmod 31 = 1$
 - $\text{inv}(10, 87) = 61$ Ya que $61 \cdot 10 \bmod 87 = 610 \bmod 87 = 7 \cdot 87 + 1 \bmod 87 = 1$
 - Lógicamente $\text{inv}(15, 87)$ no existe porque $\text{MCD}(15, 87) = 3$

Inversos aditivo y multiplicativo en primos

(A+B) mod 5						(A*B) mod 5					
+ B	0	1	2	3	4	* B	0	1	2	3	4
A 0	0	1	2	3	4	A 0	0	0	0	0	0
1	1	2	3	4	<u>0</u>	1	0	1	2	3	4
2	2	3	4	<u>0</u>	1	2	0	2	4	<u>1</u>	3
3	3	4	<u>0</u>	1	2	3	0	3	<u>1</u>	4	2
4	4	<u>0</u>	1	2	3	4	0	4	3	2	<u>1</u>

$0+0 = 0$
 $1*1 = 1$
 Son triviales

- En la operación suma siempre existirá el inverso que entregue el valor de identidad de la adición (0) para cualquier resto. El inverso de 0 será lógicamente 0
- En la operación producto, existirá el inverso que entregue el valor de identidad de la multiplicación (1) cuando se cumple que el número y el módulo sean primos entre sí. Si el módulo es primo, todos los restos, excepto el 0, tienen inverso. El inverso multiplicativo de 1, independientemente del módulo, es obviamente 1

Inverso multiplicativo número compuesto

$(A*B) \bmod 10$

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	0	2	4	6	8
3	3	6	9	2	5	8	<u>1</u>	4	7
4	4	8	2	6	0	4	8	2	6
5	5	0	5	0	5	0	5	0	5
6	6	2	8	4	0	6	2	8	4
7	7	4	<u>1</u>	8	5	2	9	6	3
8	8	6	4	2	0	8	6	4	2
9	9	8	7	6	5	4	3	2	<u>1</u>

Para módulo 10 sólo encontramos inversos multiplicativos en los restos 3, 7 y 9, puesto que los demás restos tienen factores 2 y 5 en común con el módulo ($10 = 2*5$)

Para $x = 3$, $x = 7$ y $x = 9$ se cumple que:

$$\text{MCD}(x, 10) = 1$$

$$\text{inv}(3, 10) = 7$$

$$\text{inv}(7, 10) = 3$$

$$\text{inv}(9, 10) = 9$$

Comprobación de existencia $\text{inv}(a, n)$

- \exists inverso a^{-1} en mod n ssi $\text{MCD}(a, n) = 1$
- Si $\text{MCD}(a, n) = 1$, el resultado de $a*i \bmod n$ (para i todos los restos de n) serán valores distintos dentro de n

$$\text{MCD}(a, n) = 1 \Rightarrow \exists x! \ 0 < x < n \ / \ a * x \bmod n = 1$$

- Sea: $a = 4$ y $n = 9$ Valores de $i = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- Comprobando que $\text{inv}(4, 9)$ existe y cálculo por fuerza bruta:

Como $\text{MCD}(4, 9) = 1$, sí existe el inverso

$$4*1 \bmod 9 = 4 \quad 4*2 \bmod 9 = 8 \quad 4*3 \bmod 9 = 3 \quad 4*4 \bmod 9 = 7$$

$$4*5 \bmod 9 = 2 \quad 4*6 \bmod 9 = 6 \quad \mathbf{4*7 \bmod 9 = 1} \quad 4*8 \bmod 9 = 5$$

- Existe una única solución $\text{inv}(4, 9) = 7$ ya que $4x7 = 28 \bmod 9 = 1$

Comprobación de no existencia inv (a, n)

- Si $\text{MCD}(a, n) \neq 1 \Rightarrow \text{No } \exists x ! 0 < x < n / a * x \bmod n = 1$
- Sea: $a = 3$ y $n = 15$ Valores de $i = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$
- Comprobando que inv (3, 15) no existe y cálculo por fuerza bruta:

Como $\text{MCD}(3, 15) = 3$, no existe el inverso

$3*1 \bmod 15 = 3$	$3*2 \bmod 15 = 6$	$3*3 \bmod 15 = 9$	$3*4 \bmod 15 = 12$
$3*5 \bmod 15 = 0$	$3*6 \bmod 15 = 3$	$3*7 \bmod 15 = 6$	$3*8 \bmod 15 = 9$
$3*9 \bmod 15 = 12$	$3*10 \bmod 15 = 0$	$3*11 \bmod 15 = 3$	$3*12 \bmod 15 = 6$
$3*13 \bmod 15 = 9$	$3*14 \bmod 15 = 12$		

- No existe el inv (3, 15) puesto que $\text{MCD}(3, 15) = 3$

Existencia y unicidad de $x = \text{inv}(a, n)$

- Existirá un único x tal que $x \cdot a \bmod n = 1$, es decir $x = a^{-1}$
- Ejemplos para $p = 13$ (primo) y $n = 15 = 3 \cdot 5$ (compuesto) y restos 2 y 3

$2 \cdot 0 \bmod 13 = 0$	$2 \cdot 1 \bmod 13 = 2$	$2 \cdot 2 \bmod 13 = 4$	$2 \cdot 3 \bmod 13 = 6$	$2 \cdot 4 \bmod 13 = 8$
$2 \cdot 5 \bmod 13 = 10$	$2 \cdot 6 \bmod 13 = 12$	$2 \cdot 7 \bmod 13 = 1$	$2 \cdot 8 \bmod 13 = 3$	$2 \cdot 9 \bmod 13 = 5$
$2 \cdot 10 \bmod 13 = 7$	$2 \cdot 11 \bmod 13 = 9$	$2 \cdot 12 \bmod 13 = 11$		
$3 \cdot 0 \bmod 13 = 0$	$3 \cdot 1 \bmod 13 = 3$	$3 \cdot 2 \bmod 13 = 6$	$3 \cdot 3 \bmod 13 = 9$	$3 \cdot 4 \bmod 13 = 12$
$3 \cdot 5 \bmod 13 = 2$	$3 \cdot 6 \bmod 13 = 5$	$3 \cdot 7 \bmod 13 = 8$	$3 \cdot 8 \bmod 13 = 11$	$3 \cdot 9 \bmod 13 = 1$
$3 \cdot 10 \bmod 13 = 4$	$3 \cdot 11 \bmod 13 = 7$	$3 \cdot 12 \bmod 13 = 10$		

Para el módulo primo $p = 13$, cualquier resto a tendrá inverso pues que se cumplirá siempre que $\text{MCD}(a, p) = 1$; es decir, a y p son coprimos

$2 \cdot 0 \bmod 15 = 0$	$2 \cdot 1 \bmod 15 = 2$	$2 \cdot 2 \bmod 15 = 4$	$2 \cdot 3 \bmod 15 = 6$	$2 \cdot 4 \bmod 15 = 8$
$2 \cdot 5 \bmod 15 = 10$	$2 \cdot 6 \bmod 15 = 12$	$2 \cdot 7 \bmod 15 = 14$	$2 \cdot 8 \bmod 15 = 1$	$2 \cdot 9 \bmod 15 = 3$
$2 \cdot 10 \bmod 15 = 5$	$2 \cdot 11 \bmod 15 = 7$	$2 \cdot 12 \bmod 15 = 9$	$2 \cdot 13 \bmod 15 = 11$	$2 \cdot 14 \bmod 15 = 13$
$3 \cdot 0 \bmod 15 = 0$	$3 \cdot 1 \bmod 15 = 3$	$3 \cdot 2 \bmod 15 = 6$	$3 \cdot 3 \bmod 15 = 9$	$3 \cdot 4 \bmod 15 = 12$
$3 \cdot 5 \bmod 15 = 0$	$3 \cdot 6 \bmod 15 = 3$	$3 \cdot 7 \bmod 15 = 6$	$3 \cdot 8 \bmod 15 = 9$	$3 \cdot 9 \bmod 15 = 12$
$3 \cdot 10 \bmod 15 = 0$	$3 \cdot 11 \bmod 15 = 3$	$3 \cdot 12 \bmod 15 = 6$	$3 \cdot 13 \bmod 15 = 9$	$3 \cdot 14 \bmod 15 = 12$

Para el módulo de un número compuesto $n = 15$, sólo tendrán inversos aquellos restos a que cumplan con $\text{MCD}(a, n) = 1$

Inversos mod 27 en la criptografía clásica

x	inv (x, 27)	x	inv (x, 27)	x	inv (x, 27)
1	1	10	19	19	10
2	14	11	5	20	23
4 \longrightarrow	7	13	25	22	16
5	11	14	2	23	20
7 \longrightarrow	4	16	22	25	13
8	17	17	8	26	26

- $27 = 3^3$ luego no existe inverso para $x = 3, 6, 9, 12, 15, 18, 21, 24$
- Si $\text{inv}(x, n) = a$, entonces $\text{inv}(a, n) = x$
- $\text{inv}(1, n) = 1$
- $\text{inv}(n-1, n) = n-1$

Inversos en la criptografía moderna

- Algoritmo IDEA

El algoritmo IDEA usa una operación producto modular mod 65.537 ($2^{16} + 1$) que es un número primo, con la clave Z1 de 16 bits y la primera palabra de 16 bits del bloque de entrada de 64 bits. En el descifrado, en esa zona se usará el inverso multiplicativo. Si $Z1 = 6.018$, ¿cuál es el valor de ese inverso $Z1^{-1}$?
Solución: $Z1^{-1} = \text{inv}(6.018, 65.537) = 18.045$

- Algoritmo RSA

Si $p = 761$ y $q = 919$, siendo la clave pública $e = 65.537$, encuentra la clave privada d si se sabe que $d = \text{inv}(e, \phi(n))$
Solución: $d = \text{inv}[e, (p-1)(q-1)] = \text{inv}[65.537, 760 \cdot 918]$
 $d = \text{inv}[65.537, 697.680] = 379.313$

Operaciones sencillas con



https://www.criptored.es/software/sw_m001t.htm

Inverso de a en p por el teorema de Fermat

- El pequeño teorema de Fermat dice lo siguiente:
 - Si el cuerpo de trabajo es un primo p
 - $\text{MCD}(a, p) = 1 \Rightarrow a^{\phi(p)} \bmod p = 1$
 - Entonces: $a * x \bmod p = 1$
 - Y además: $a^{\phi(n)} \bmod p = 1$
 - Además, en este caso $\phi(p) = p-1$ por lo que igualando las dos ecuaciones con resultado igual a 1 de arriba tenemos:
 - $a * x \bmod p = a^{\phi(n)} \bmod p$
 - $a^{\phi(p)} * a^{-1} \bmod p = x \bmod p$
 - $x = a^{p-2} \bmod p$
 - Luego, x será el inverso de a en el primo p

Ejemplo:
 $x = \text{inv}(8, 11)$
 $x = 8^{11-2} \bmod 11 = 8^9 \bmod 11$
 $x = 134.217.728 \bmod 11 = 7$
Efectivamente, se cumple que:
 $8 * 7 \bmod 11 = 56 \bmod 11 = 1$

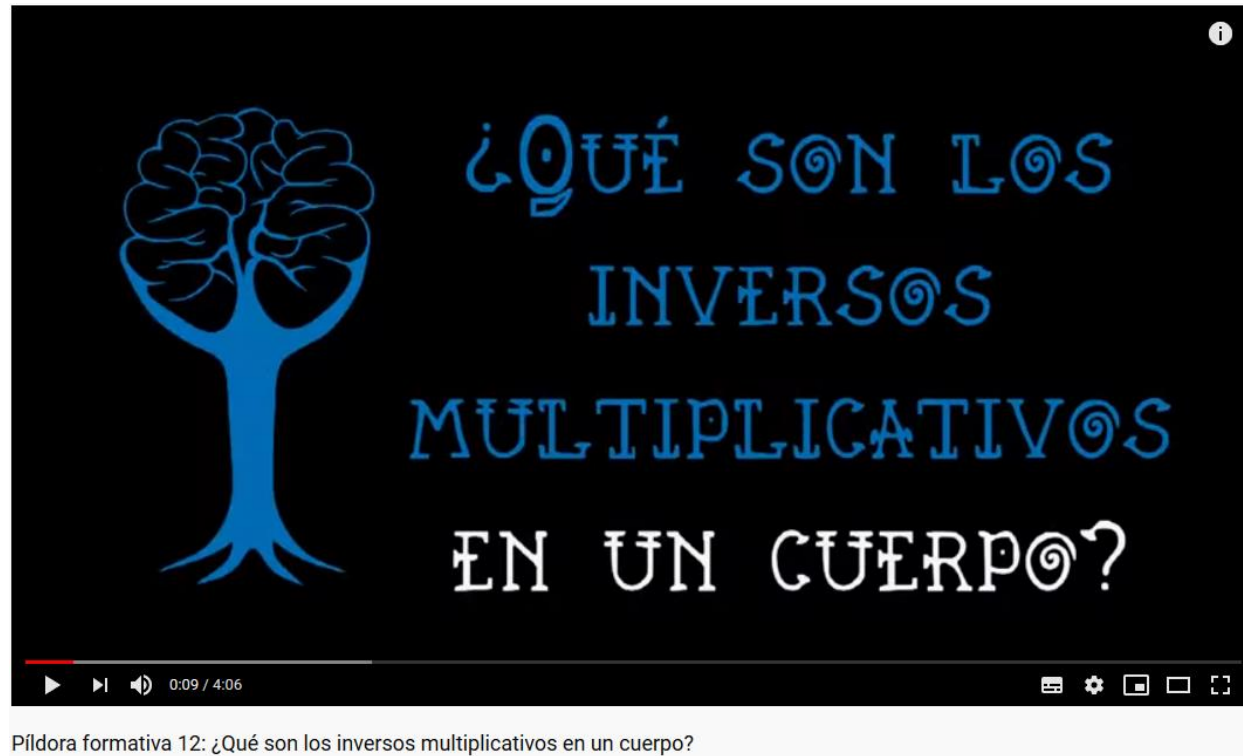
Inverso de a en n por el teorema de Euler

- Si $\text{MCD}(a, n) = 1 \Rightarrow a^{\phi(n)} \bmod n = 1$
- Ahora igualamos $a * x \bmod n = 1$ y $a^{\phi(n)} \bmod n = 1$
 $a^{\phi(n)} * a^{-1} \bmod n = x \bmod n$ por lo tanto: $x = a^{\phi(n)-1} \bmod n$
- El valor x será el inverso de a en n. En el cálculo anterior se ha realizado una operación multiplicando por a^{-1} , que es similar a la *división* por a. Esto se puede hacer porque $\text{MCD}(a, n) = 1$ y por lo tanto hay un único valor inverso en n que lo permite.
- Ejemplo: si $n = 10$ y $a = 3$, se cumple que $\text{MCD}(3, 10) = 1$ y $\phi(10) = 4$
Entonces $x = a^{\phi(n)-1} \bmod n = 3^{4-1} \bmod 10 = 3^3 \bmod 10 = 27 \bmod 10 = 7$
Efectivamente $\text{inv}(3, 10) = 7$ porque $3 * 7 = 21 \bmod 10 = 1$

¿Y si no conocemos $\phi(n)$?

- Calcular $a^i \bmod n$, cuando los valores de i y a son grandes, se hace tedioso pues habría que utilizar la reducción por cuadrados repetidas veces
- No obstante, hay otros métodos mucho más eficientes y rápidos para hacer esos cálculos de exponenciación como por ejemplo el Algoritmo de Exponenciación Rápida AER, que veremos en una próxima clase
- Además, si no conocemos $\phi(n)$ o no queremos usar los teoremas de Euler o de Fermat antes vistos, siempre podremos encontrar el inverso de a en n usando el Algoritmo Extendido de Euclides AEE, que veremos en una próxima clase, y que será el método más recomendado y eficiente

Más información en píldoras Thoth



<https://www.youtube.com/watch?v=MMRNsZ28IIs>

Conclusiones de la lección 2.3

- El inverso aditivo $\text{inv}_+(a, n)$ siempre existirá, será el complemento al módulo
- El inverso xor $\text{inv}_\oplus(a)$ siempre existirá y es el mismo número porque xor es una función involutiva
- El inverso multiplicativo $\text{inv}(a, n)$ existirá si y solo si $\text{MCD}(a, n) = 1$
- Si el módulo es primo, $n = p$, todos restos a de p tendrán inverso
- Los inversos deben ser únicos
- Se puede encontrar el inverso multiplicativo por fuerza bruta (que no es recomendable), usando el pequeño teorema de Fermat para primos o usando el teorema de Euler para números compuestos. En la práctica, se usará el Algoritmo Extendido de Euclides AEE para calcular inversos multiplicativos

Lectura recomendada

- Guion píldora formativa Thoth nº 23, ¿Qué son los inversos aditivos en un cuerpo?, Jorge Ramió, 2014
 - <https://www.criptored.es/thoth/material/texto/pildora023.pdf>
- Guion píldora formativa Thoth nº 12, ¿Qué son los inversos multiplicativos en un cuerpo?, Jorge Ramió, 2015
 - <https://www.criptored.es/thoth/material/texto/pildora012.pdf>
- Inverso multiplicativo (aritmética modular), Wikipedia
 - [https://es.wikipedia.org/wiki/Inverso_multiplicativo_\(aritm%C3%A9tica_modular\)](https://es.wikipedia.org/wiki/Inverso_multiplicativo_(aritm%C3%A9tica_modular))
- Inversos modulares, Khan Academy
 - <https://es.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-inverses>

Class4crypt c4c2.4

Módulo 2. Matemáticas discretas en la criptografía

Lección 2.4. Cálculo de inversos con el algoritmo extendido de Euclides

2.4.1. Recordando la importancia de los inversos en criptografía

2.4.2. Divisibilidad de los números y Euclides

2.4.3. Algoritmo de Euclides para calcular el máximo común divisor

2.4.4. Cálculo de inversos mediante el algoritmo de Euclides y el recorrido de la tabla de restos

2.4.5. Algoritmo extendido de Euclides para el cálculo de inversos

Class4crypt c4c2.4 Cálculo de inversos con el algoritmo extendido de Euclides

<https://www.youtube.com/watch?v=fYVDUAQ1R-o>

Recordando los inversos en criptografía

- Los números inversos son muy importantes en criptografía porque en muchos casos permiten descifrar en destino lo que se ha cifrado en origen
- Existen inversos aditivos, inversos xor e inversos multiplicativos, siendo estos últimos los más interesantes puesto que dan lugar al concepto de clave pública y clave privada, inversas entre sí, de la criptografía asimétrica
- Cuando se indica sólo $\text{inv}(a, n)$, se entiende que es un inverso multiplicativo
- Los inversos en algoritmos de cifra asimétrica como RSA y Elgamal, no son tan directos como los inversos de una cifra por multiplicación simple
- Para encontrar los inversos multiplicativos, siempre se usará el algoritmo extendido de Euclides, representado comúnmente por las letras AEE. La pregunta es: ¿por qué usamos Euclides para encontrar los inversos?

Divisibilidad de los números y Euclides

- En criptografía muchas veces nos interesará encontrar el máximo común divisor entre dos números a y b , esto es $\text{MCD}(a, b)$
- Para la existencia del inverso del resto a en un módulo n , ese resto a y el módulo n deberán ser coprimos, es decir debe cumplirse que $\text{MCD}(a, n) = 1$
- El algoritmo de Euclides de la divisibilidad de los números nos decía
 - a) Si x divide a a y b $a = x \cdot a'$ $b = x \cdot b'$
 - b) Por lo tanto $a - k \cdot b = x \cdot a' - k \cdot x \cdot b'$
 - c) Es decir $a - k \cdot b = x(a' - k \cdot b')$ Se concluye que x divide a $(a - k \cdot b)$

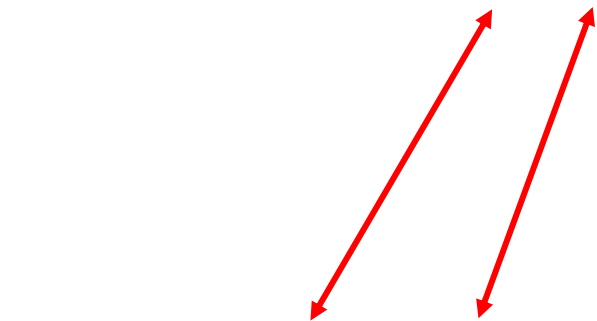
Si $a = 32$, $b = 12$ y $x = 4$ [$a = x \cdot a'$ ($32 = 4 \cdot 8$)] con $a' = 8$ y $b = x \cdot b'$ [$12 = 4 \cdot 3$] con $b' = 3$

Entonces x divide a $(a - k \cdot b)$ es decir 4 divide a $(32 - k \cdot 12)$. Se cumple para todo valor de k :

Para $k = \dots, -3, -2, -1, 0, 1, 2, 3, \dots$ se obtiene $(68, 56, 44, 32, 20, 8, -4)$ todos divisibles por 4

El máximo común divisor MCD

- Como hemos llegado a que x divide a $(a - k*b)$, esto nos permitirá encontrar el máximo común divisor entre a y b
- Si $a > b$ (28 y 20) entonces $a = d_1 * b + r$ Ejemplo $28 = 1 * 20 + 8$
- Con d_i un entero (1) y r (8) un resto de b
- Luego $\text{MCD}(a, b) = \text{MCD}(b, r)$ Con $a > b > r \geq 0$
- Porque:
- Si $b > r$ (20 y 8) $b = d_2 * r + r'$ Ejemplo $20 = 2 * 8 + 4$
- Con r ahora un entero (2) y r' (4) un resto r
- Esto se puede ir repitiendo en el cálculo de MCD hasta obtener un resto 0; en este caso, $\text{MCD}(28, 20) = 4$



Divisibilidad con el algoritmo de Euclides

$$\begin{aligned} \text{MCD}(148, 40) \\ 148 &= 3 * 40 + 28 \\ 40 &= 1 * 28 + 12 \\ 28 &= 2 * 12 + 4 \\ 12 &= 3 * 4 + 0 \\ \text{MCD}(148, 40) &= 4 \end{aligned}$$

Esta condición
será importante
en criptografía

$$\begin{aligned} 148 &= 2^2 * 37 \\ 40 &= 2^3 * 5 \\ \text{Factor común} \\ 2^2 &= 4 \end{aligned}$$

No hay
factores
en común

$$\begin{aligned} 385 &= 5 * 7 * 11 \\ 78 &= 2 * 3 * 13 \end{aligned}$$

$$\begin{aligned} \text{MCD}(385, 78) \\ 385 &= 4 * 78 + 73 \\ 78 &= 1 * 73 + 5 \\ 73 &= 14 * 5 + 3 \\ 5 &= 1 * 3 + 2 \\ 3 &= 1 * 2 + 1 \\ 2 &= 2 * 1 + 0 \\ \text{MCD}(385, 78) &= 1 \end{aligned}$$

Los restos del algoritmo de Euclides

- Ordenando por restos
- $n = C_1 * a + r_1$ Con $a > r_1$
- $a = C_2 * r_1 + r_2$ Con $r_1 > r_2$
- $r_1 = C_3 * r_2 + r_3$ Con $r_2 > r_3$
- $r_2 = C_4 * r_3 + r_4$ Con $r_3 > r_4$
-
- $r_{n-2} = C_n * r_{n-1} + 1$ Con $r_{n-1} > 1$
- $r_{n-1} = C_{n+1} * 1 + 0$
- Concluye el algoritmo


Si volvemos hacia atrás desde este valor, obtendremos el inverso de a en el módulo n



Tabla de restos del algoritmo de Euclides

- Ordenando por restos desde el valor 1, se llega a una expresión del tipo $(k_1 * n + k_2 * a) \bmod n = 1$, en donde el inverso de a en n lo dará el coeficiente k_2 puesto que $k_1 * n \bmod n = 0$

	C_1	C_2	C_3	C_4	...	C_{n-1}	C_n	C_{n+1}
n	a	r_1	r_2	r_3	...	r_{n-2}	r_{n-1}	1



$$(k_1 * n + k_2 * a) \bmod n = 1$$

Por lo tanto: $k_2 * a \bmod n = 1$

Inversos con tabla de restos de Euclides

- Encontrar el inv (9, 25) con la tabla de restos de Euclides

$$25 = 2 \cdot 9 + 7$$

$$9 = 1 \cdot 7 + 2$$

$$7 = 3 \cdot 2 + 1$$

$$2 = 2 \cdot 1 + 0$$

$$7 = 25 - 2 \cdot 9$$

$$2 = 9 - 1 \cdot 7$$

$$1 = 7 - 3 \cdot 2$$

$$0 = 2 - 2 \cdot 1$$

Restos

$$7 = 25 - 2 \cdot 9$$

$$2 = 9 - 1 \cdot (25 - 2 \cdot 9) = 3 \cdot 9 - 1 \cdot 25$$

$$1 = (25 - 2 \cdot 9) - 3 \cdot (3 \cdot 9 - 1 \cdot 25)$$

$$1 = 4 \cdot 25 - 11 \cdot 9 \pmod{25} = -11 \cdot 9$$

Como $1 = -11 \cdot 9 \pmod{25}$

$$\text{inv}(9, 25) = -11 + 25 = 14$$

Tabla de restos

	2	1	3	2	
25	9	7	2	1	0

Más información en píldoras Thoth



Píldora formativa 24: ¿Por qué usamos el algoritmo de Euclides para calcular inversos?

<https://www.youtube.com/watch?v=LBgq4NKLHws>

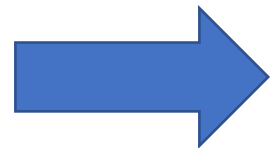
El algoritmo extendido de Euclides

- Para encontrar $x = \text{inv}(A, B)$
Hacer $(g_0, g_1, u_0, u_1, v_0, v_1, i) = (B, A, 1, 0, 0, 1, 1)$
Mientras $g_i \neq 0$ hacer
 Hacer $y_{i+1} = \text{parte entera}(g_{i-1}/g_i)$
 Hacer $g_{i+1} = g_{i-1} - y_{i+1} * g_i$
 Hacer $u_{i+1} = u_{i-1} - y_{i+1} * u_i$
 Hacer $v_{i+1} = v_{i-1} - y_{i+1} * v_i$
 Hacer $i = i+1$
Si $(v_{i-1} < 0)$
 Hacer $v_{i-1} = v_{i-1} + B$
Hacer $x = v_{i-1}$

Calculemos ahora:

$\text{inv}(4, 31)$

$\text{inv}(9, 32)$



Calculando inversos con el AEE (1/2)

- $\text{inv}(4, 31)$

i	y_i	g_i	u_i	v_i
0	--	31	1	0
1	--	4	0	1
2	7	3	1	-7
3	1	1	-1	8
4	3	0	4	-31

- $\text{inv}(4, 31) = 8$
- $4 \cdot 8 \bmod 31 = 32 \bmod 31 = 1$
- Si número de pasos = par, se obtiene $(a, -n)$

i	y_i	g_i	u_i	v_i
0	-	31	1	0
1	-	4	0	1
2	7	3	1	-7
3	1	1	-1	8
4	3	0	4	-31

Calculando inversos con el AEE (2/2)

- $\text{inv}(9, 32)$

i	y_i	g_i	u_i	v_i
0	--	32	1	0
1	--	9	0	1
2	3	5	1	-3
3	1	4	-1	4
4	1	1	2	-7
5	4	0	-9	32

- $\text{inv}(9, 32) = -7 = -7 + 32 \bmod 32 = 25$
- $25 \cdot 9 \bmod 32 = 225 \bmod 32 = 1$
- Si número de pasos = impar, se obtiene $(-a, n)$

i	y_i	g_i	u_i	v_i
0	-	32	1	0
1	-	9	0	1
2	3	5	1	-3
3	1	4	-1	4
4	1	1	2	-7
5	4	0	-9	32

Cálculo de inversos online con el AEE

Modular inversion

Use the extended Euclidean algorithm to compute a modular multiplicative inverse

Computes m for $n^{-1} = m \pmod{p}$, where n and p are coprime.

Displays the steps of the extended Euclidean algorithm.

$n =$, $p =$ **Cuidado: n = resto y p = módulo**

step	quotient (q)	t_1	t_2	t_3	u_1	u_2	u_3 (p)	v_1	v_2	v_3 (n)
0					1	0	7690	0	1	127
	$\text{floor}(u_3/v_3)$	$u_1 - q*v_1$	$u_2 - q*v_2$	$u_3 - q*v_3$						
1	60	1	-60	70	0	1	127	1	-60	70
2	1	-1	61	57	1	-60	70	-1	61	57
3	1	2	-121	13	-1	61	57	2	-121	13
4	4	-9	545	5	2	-121	13	-9	545	5
5	2	20	-1211	3	-9	545	5	20	-1211	3
6	1	-29	1756	2	20	-1211	3	-29	1756	2
7	1	49	-2967	1	-29	1756	2	49	-2967	1
8	2	-127	7690	0	49	-2967	1	-127	7690	0
mod inverse:										4723

Con el AEE normalmente el resultado converge muy rápido hacia la solución

$$\text{inv}(127, 7.690) = 4.723$$

$$4.723 * 127 = 599.821$$

$$599.821 \bmod 7.690 = 1$$

<https://dsri.github.io/modinverse/>

Darren Sri-Jayantha

Cálculo de inversos con el AEE y SAMCrypt

SAMCrypt - Software de Aritmética Modular para Criptografía

Archivo Unidades Operaciones Tablas Conversor Ayuda

Inverso

Operando: 1.234.567.890 (10 dígitos)

Módulo: 1.000.000.000.000.000.000.001 (25 dígitos)

Resultado: 407.174.901.495.291.603.607.154 (24 dígitos)

Borrar =

+ - x / √

α XOR Inv X^y Mod

MCD mcm Primalidad PFE PLD

Historial

inv(1.234.567.890 ,
1.000.000.000.000.000.000.001) =
407.174.901.495.291.603.607.154

Borrar Historial

https://www.criptored.es/software/sw_m001t.htm

Mucho cuidado con páginas Web (errores)

NUM 1: 2346527345000000056878; NUM 2: 89188191800000008911179 (ver URL en lectura extra)

Algoritmo de Euclides extendido

Primer número 2346527345000000	Segundo número 8918819180000000
Calcular	Borrar
Máximo común divisor 262144	
Que puede expresarse como $(-4213637481062840) \cdot 8.918819180000002e+22$	
El inverso de 2.346527345e módulo 8.9188191800 es No existe.	

← FALSO

← FALSO

Máximo Común Divisor

Operando 1: 2.346.527.345.000.000.056.878 (22 dígitos)

Operando 2: 89.188.191.800.000.008.911.179 (23 dígitos)

Operando 3:

Resultado: 1

← correcto

Historial

MCD(2.346.527.345.000.000.056.878 , 89.188.191.800.000.008.911.179) = 1

Inverso

Operando: 2.346.527.345.000.000.056.878 (22 dígitos)

Módulo: 89.188.191.800.000.008.911.179 (23 dígitos)

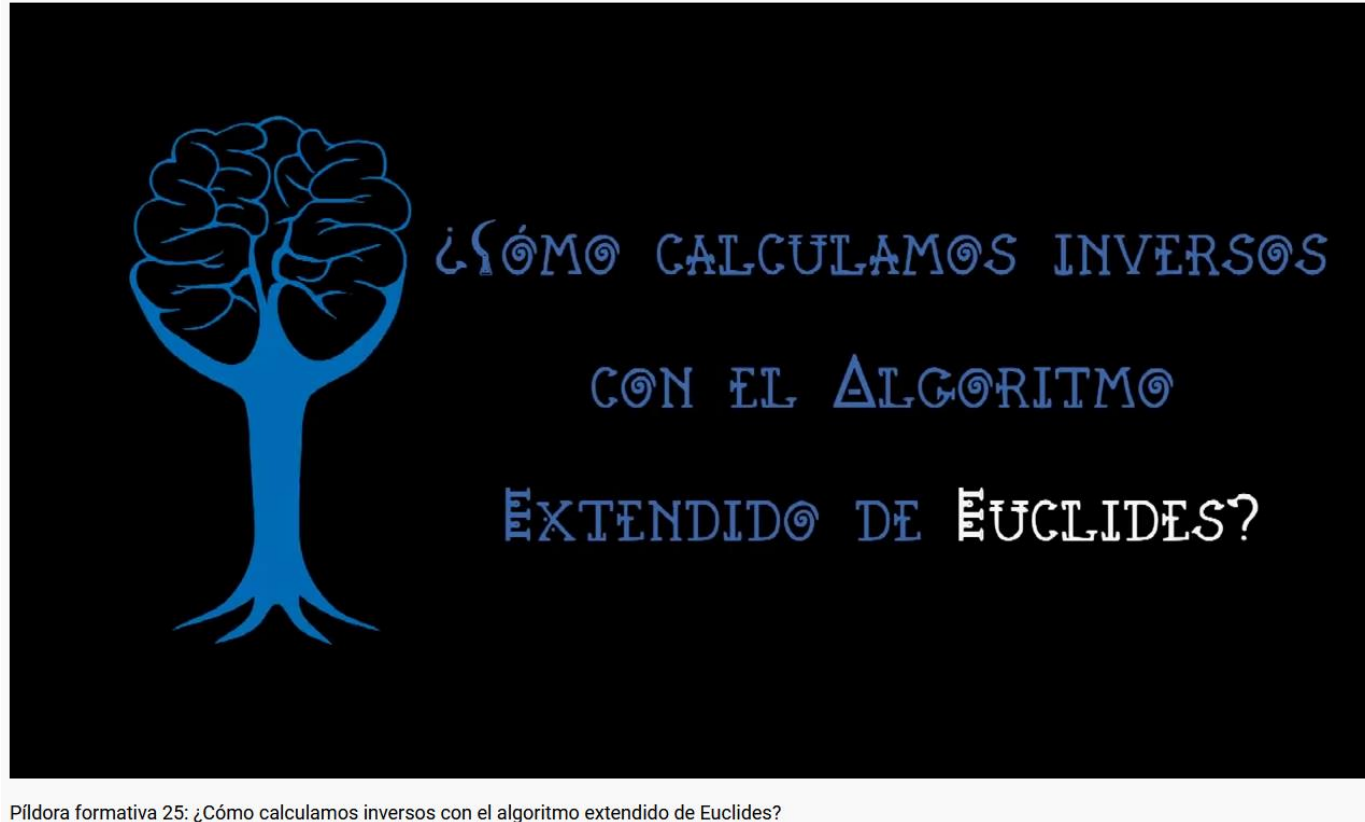
Resultado: 24.417.938.781.704.652.185.289

← correcto

Historial

inv(2.346.527.345.000.000.056.878 , 89.188.191.800.000.008.911.179) = 24.417.938.781.704.652.185.289

Más información en píldoras Thoth



<https://www.youtube.com/watch?v=D289EF58Yrw>

Conclusiones de la lección 2.4

- La operación realizada con el algoritmo de Euclides para encontrar el máximo común divisor entre dos números a y n nos permitirá encontrar el inverso del resto a en el módulo n , siempre y cuando este inverso exista
- Ordenando por restos los resultados parciales del algoritmo de Euclides, y haciendo un recorrido inverso del mismo desde el valor 1 encontrado antes de concluir, podemos encontrar el inverso del resto a en el módulo n
- Este procedimiento se conoce como algoritmo extendido de Euclides AEE, y nos permite encontrar el inverso de un resto a dentro de un módulo n de una forma muy rápida y eficiente
- El número de pasos que realiza el algoritmo extendido de Euclides hasta encontrar $\text{inv}(a, n)$, independiente de los valores del resto a y del módulo n , por lo general está por debajo de la docena

Lectura recomendada

- Guion píldora formativa Thoth nº 24, ¿Por qué usamos el algoritmo de Euclides para calcular inversos?, Jorge Ramió, 2015
 - <https://www.criptored.es/thoth/material/texto/pildora024.pdf>
- Guion píldora formativa Thoth nº 25, ¿Cómo calculamos inversos con el algoritmo extendido de Euclides?, Jorge Ramió, 2015
 - <https://www.criptored.es/thoth/material/texto/pildora025.pdf>
- Algoritmo de Euclides, Wikipedia
 - https://es.wikipedia.org/wiki/Algoritmo_de_Euclides
- Mucho cuidado con páginas Web que entregan resultados falsos, como en el ejemplo mostrado en diapositiva 19
 - <http://mimosa.pntic.mec.es/jgomez53/javas/euclidesext.htm>

Class4crypt c4c2.5

Módulo 2. Matemáticas discretas en la criptografía

Lección 2.5. Algoritmo de exponenciación modular rápida

2.5.1. Operaciones de potencia modular en criptografía moderna

2.5.2. La solución del homomorfismo de los enteros

2.5.3. Algoritmo de exponenciación rápida

2.5.4. Calculadoras modulares básicas, avanzadas y online

2.5.5. Calculadoras modulares online no recomendables

Class4crypt c4c2.5 Algoritmo de exponenciación modular rápida

<https://www.youtube.com/watch?v=sHdVYLA8XtI>

Potencia modular en criptografía

- Varios algoritmos de cifra y firma digital en criptografía asimétrica usan operaciones de exponenciación modular con números muy grandes
- Diffie y Hellman
 - $X_A = \alpha^a \bmod p$ (con p un primo, α una raíz primitiva de p y a una clave secreta de A)
 - $X_B = \alpha^b \bmod p$ (con p un primo, α una raíz primitiva de p y b una clave secreta de B)
- RSA
 - $X = K^{e_R} \bmod n_R$ (con $n_R = p_R * q_R$, e_R clave pública, R = Receptor)
 - $X = h(M)^{d_E} \bmod n_E$ (con $n_E = p_E * q_E$, d_E clave privada, E = Emisor)
- Elgamal
 - Clave privada $C_{\text{priv}} = x$
 - Clave pública $C_{\text{públ}} = \alpha^x \bmod p$ (con p un primo y α una raíz primitiva de p)

Solución: homomorfismo de los enteros

- Hemos visto en una lección anterior que no podemos realizar la operación de potencia y, posteriormente, realizar la reducción a módulo
 - No hay espacio físico donde almacenar esa inmensa cantidad de bits
 - El tiempo de cómputo es inmenso y se vuelve impracticable
- La solución estaba en el homomorfismo de los enteros
 - Las operaciones se realizaban por partes, reduciendo siempre a módulo antes de realizar la siguiente operación
 - Una primera opción es usar el método de los cuadrados, reduciendo a módulo en cada cómputo, pero no es adecuado para números grandes
 - Una segunda opción, esta ya óptima, es usar el método de exponenciación modular rápida (*square and multiply*) en el que se recorre el exponente representado en binario, haciendo operaciones diferentes si el bit es un 0 o un 1

Algoritmo de Exponenciación Rápida AER

Hallar $x = A^B \bmod n$

- Obtener representación binaria del exponente B de k bits:

$$B_2 \rightarrow b_{k-1}b_{k-2}\dots b_i\dots b_1b_0$$

- Hacer $x = 1$
- Para $i = k-1, \dots, 0$ hacer
 $x = x^2 \bmod n$
 Si $b_i = 1$ entonces
 $x = x * A \bmod n$

Ejemplo: $19^{83} \bmod 91 = 24$ ←

$$19^{83} = 1,369458509879505101557376746718e+106$$

$$83_2 = b_6b_5b_4b_3b_2b_1b_0 = 1010011$$

$$x = 1$$

$$i = 6 \quad b_6 = 1 \quad x = 1^2 * 19 \bmod 91 = 19 \quad x = 19$$

$$i = 5 \quad b_5 = 0 \quad x = 19^2 \bmod 91 = 88 \quad x = 88$$

$$i = 4 \quad b_4 = 1 \quad x = 88^2 * 19 \bmod 91 = 80 \quad x = 80$$

$$i = 3 \quad b_3 = 0 \quad x = 80^2 \bmod 91 = 30 \quad x = 30$$

$$i = 2 \quad b_2 = 0 \quad x = 30^2 \bmod 91 = 81 \quad x = 81$$

$$i = 1 \quad b_1 = 1 \quad x = 81^2 * 19 \bmod 91 = 80 \quad x = 80$$

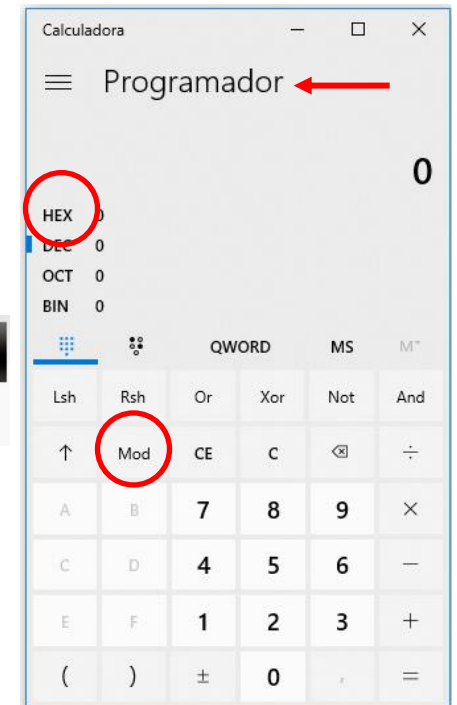
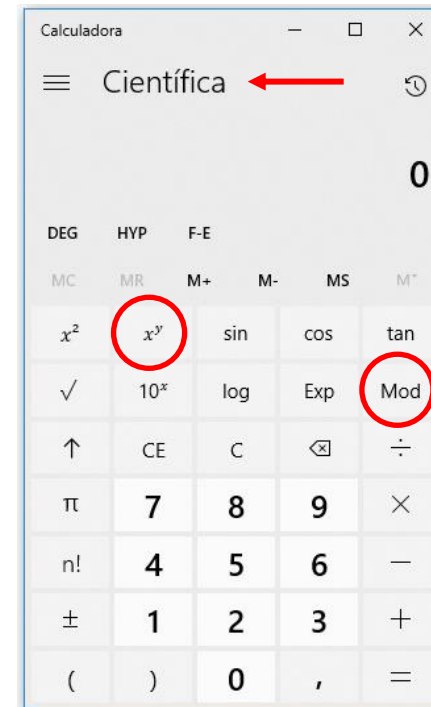
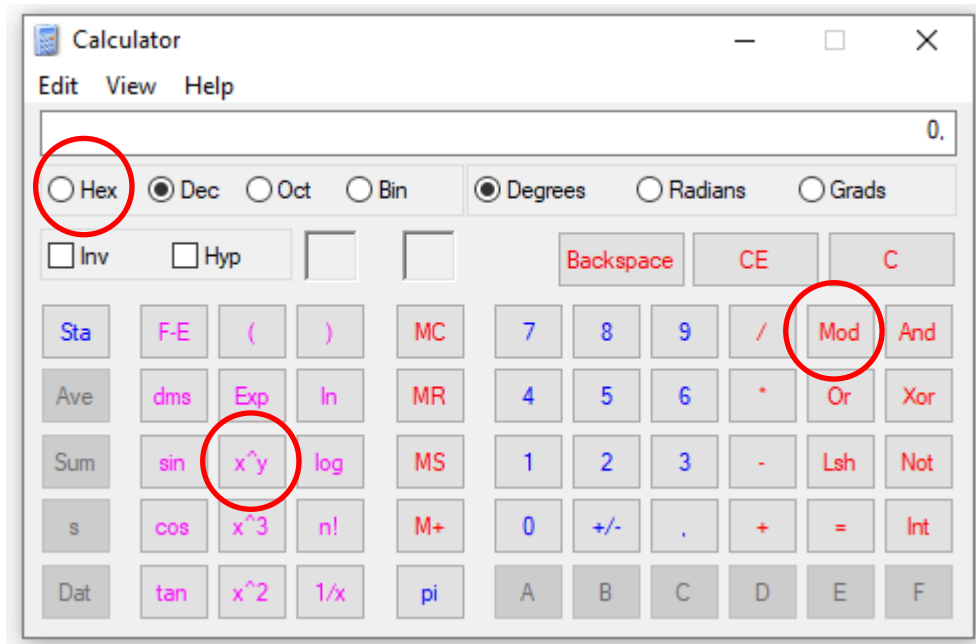
$$i = 0 \quad b_0 = 1 \quad x = 80^2 * 19 \bmod 91 = 24 \quad x = 24$$

Hemos realizado sólo 16 operaciones, frente a las 164 si se usase reducción por cuadrados.

¿Qué sucederá en una firma digital RSA sobre un hash: $(256 \text{ bits})^{(2.048 \text{ bits})} \bmod (2.048 \text{ bits})$?

Con AER, 2.048 cuadrados y unas 1.000 multiplicaciones... pero ¿un número de 620 dígitos?

Calculadoras modulares básicas



Calculadoras de Windows: 5 dígitos^{5 dígitos} ya indica “entrada no válida”

Calculadoras modulares avanzadas

SAMCrypt - Software de Aritmética Modular para Criptografía

Archivo Unidades Operaciones Tablas Conversor Ayuda

Potencia

Base: 1.234.981.234.192.846.389.126.438.928.463.891.264.38
1000 dígitos

Exponente: 72.307.642.305.462.346.508.234.650.823.645.823.560.3
2000 dígitos

☒ Módulo: 5.432.085.603.248.762.384.560.238.745.602.837.456.08
1000 dígitos

Resultado: 3.971.090.396.861.519.135.217.031.533.201.742.908.85 D 1000 dígitos

Borrar =

+	-	X	/	√
α	XOR	Inv	X^y	Mod
MCD	mcm	Primalidad	PFE	PLD

Historial

1.234.981.234.192.846.389.126.438.928.463.891
.264.389.284.638.912.643.892.846.389.126.438.
928.463.891.264.389.284.638.912.643.892.846.3
89.126.438.928.463.891.264.389.284.638.912.64
3.892.846.389.126.438.928.463.891.264.389.284
.638.912.643.892.846.389.126.438.928.463.891.
264.389.284.638.912.643.892.846.389.126.438.9
28.463.891.264.389.284.638.912.643.892.846.38
9.126.438.928.463.891.264.389.284.638.912.643
.892.846.389.126.438.928.463.891.264.389.284.
638.912.643.892.846.389.126.438.928.463.891.2
64.389.284.638.912.643.892.846.389.126.438.92
8.463.891.264.389.284.638.912.643.892.846.389
.126.438.928.463.891.264.389.284.638.912.643.
892.846.389.126.438.928.463.891.264.389.284.6
38.912.643.892.846.389.126.438.928.463.891.26
4.389.284.638.912.643.892.846.389.126.438.928
.463.891.264.389.284.638.912.643.892.846.389.
126.438.928.463.891.264.389.284.638.912.643.8
92.846.389.126.438.928.463.891.264.389.284.63
8.912.643.892.846.389.126.438.928.463.891.264
.389.284.638.912.643.892.846.389.126.438.928.

Borrar Historial



https://www.criptored.es/software/sw_m001t.htm

Calculadoras online

MOBILEFISH.COM

Big number equation calculation

https://www.mobilefish.com/services/big_number_equation/big_number_equation.php

No resulta muy cómodo

1.000 dígitos^{2.000 dígitos} mod 1.000 dígitos
lo calcula en 1 segundo, como SAMCrypt

<https://www.boxentriq.com/code-breaking/modular-exponentiation>

[HOME](#)
[ABOUT](#)
[CODE BREAKING](#)
[FAQ](#)
[CONTACT](#)

Modular Exponentiation Calculator

Free and fast online Modular Exponentiation (ModPow) calculator. Just type in the base number, exponent and modulo, and click Calculate. This [Modular Exponentiation](#) calculator can handle big numbers, with any number of digits, as long as they are positive integers.

For a more comprehensive mathematical tool, see the [Big Number Calculator](#).

-> Try your skills on Boxentriq's puzzles. Click here. <-

Calculate $a^b \bmod m$

Number (a)

1234981234192846389126438928463891264389

284638912643892846389126438928463891264

389284638912643892846389126438928463891

☐ Use hexadecimal numbers

Calculate

Exponent (b)

72307642305462346508234650823645823560

327465087234650823456283756873658073465

87643875608324687608678068876510874357

Modulo (m)

54320856032487623845602387456028374560

82374560832756082734560827345068732605

872346587263485620348756203847560283745

Result

3971090396861519135217031533201742908850055928778611176729052103058641886822417723738199645333669149824313757246927954742621208563452843182145

89402942365834299942719364481942125960371354272557418206764009723107902746000807196412614862880337032572345653574528846806981188723454582

9644358905295584931797140208383028402646041913458306526732307814083101766573030000303127980765927939189967655762643010155287907327124283401

Calculadoras online no recomendables I

MATH 139

PowerMod Calculator
Computes $(\text{base})^{(\text{exponent})} \bmod (\text{modulus})$ in $\log(\text{exponent})$ time.

Base: 1234567890	Exponent: 9876543210	Modulus: 78120
Compute	$b^e \bmod m =$	56512

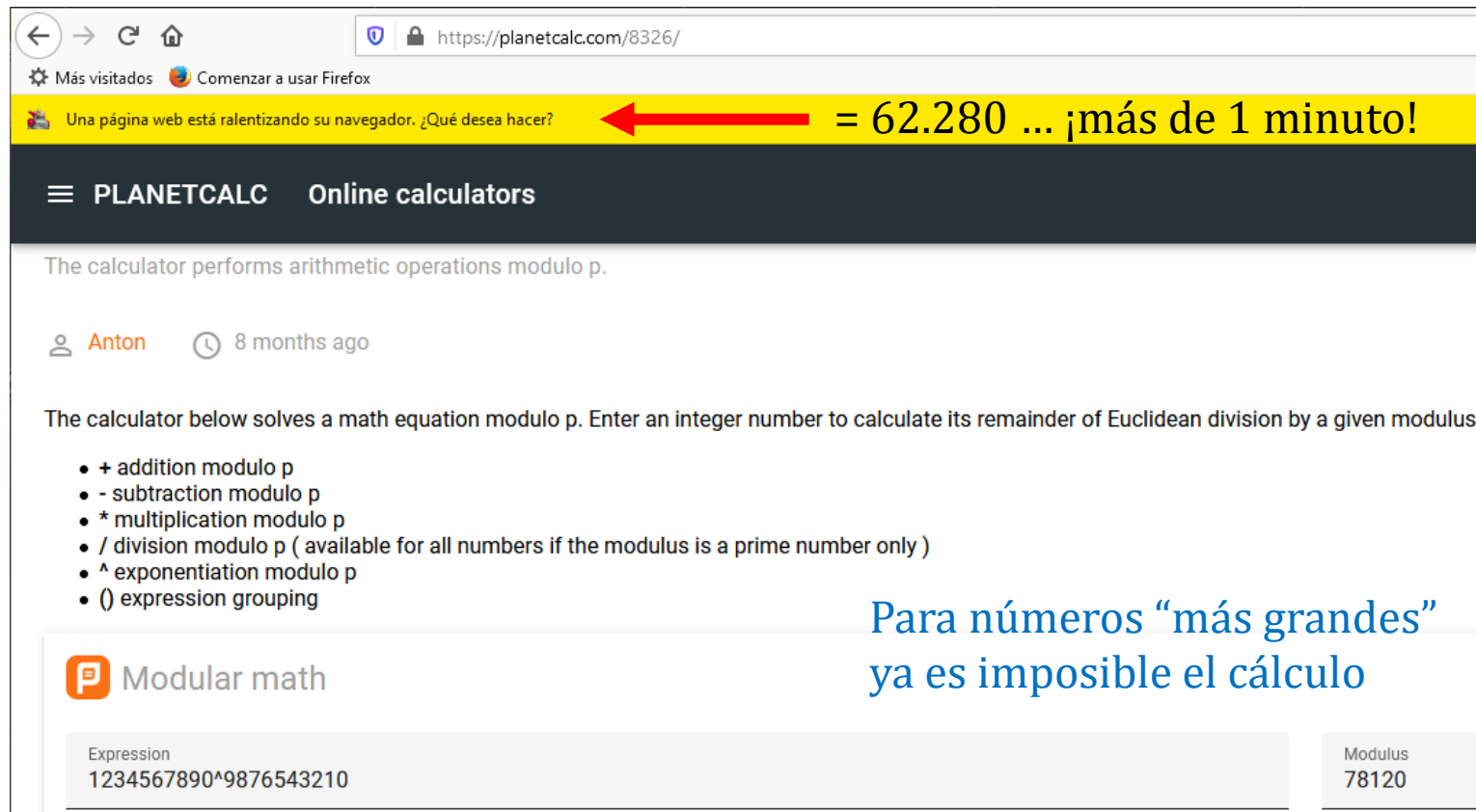
The program is written in JavaScript, and runs on the client computer. Most implementations seem to handle numbers of up to 16 digits correctly.

Resultado **falso**: $1234567890^{9876543210} \bmod 78120 = 62.280$

Y para números grandes, se bloquea el navegador...

<https://www.mtholyoke.edu/courses/quenell/s2003/ma139/js/powermod.html>

Calculadoras online no recomendables II



The screenshot shows a web browser window with the URL <https://planetcalc.com/8326/>. A yellow warning bar at the top states: "Una página web está ralentizando su navegador. ¿Qué desea hacer?" with a red arrow pointing left and the text "= 62.280 ... ¡más de 1 minuto!". Below the warning, the page header reads "PLANETCALC Online calculators". The main content area describes a modular calculator that performs arithmetic operations modulo p. It lists the supported operations: addition, subtraction, multiplication, division, exponentiation, and expression grouping. The calculator interface shows the expression $1234567890^{9876543210} \bmod 78120$ and the result 62.280.

Una página web está ralentizando su navegador. ¿Qué desea hacer? ← = 62.280 ... ¡más de 1 minuto!

PLANETCALC Online calculators

The calculator performs arithmetic operations modulo p.

Anton 8 months ago

The calculator below solves a math equation modulo p. Enter an integer number to calculate its remainder of Euclidean division by a given modulus.

- + addition modulo p
- subtraction modulo p
- * multiplication modulo p
- / division modulo p (available for all numbers if the modulus is a prime number only)
- ^ exponentiation modulo p
- () expression grouping

Modular math

Expression: $1234567890^{9876543210}$ Modulus: 78120

Para números “más grandes”
ya es imposible el cálculo

$$1234567890^{9876543210} \bmod 78120 = 62.280$$

<https://planetcalc.com/8326/>

Más información en píldoras Thoth



<https://www.youtube.com/watch?v=atadETMx9Lk>

Conclusiones de la lección 2.5

- La criptografía moderna usa en sus algoritmos de clave pública operaciones de exponenciación modular $A^B \bmod n$ con números muy grandes, para protegerse de ataques por fuerza bruta
- No obstante, para permitir realizar las operaciones de cifrado y descifrado en tiempos mínimos, se hace uso del homomorfismo de los enteros y, en particular, del algoritmo de exponenciación rápida AER
- El exponente B se expresa en binario y se recorre dicha cadena de bits, de forma que cuando el bit es un 0 la operación a realizar es un cuadrado, y cuando el bit es un 1, además de ese cuadrado se multiplica por la base A , en ambos casos reduciendo la operación módulo n
- Operaciones en las que tanto A , B y n son mayores que 3.000 bits, se realizan en menos de un segundo... y cuidado nuevamente con páginas web no fiables

Lectura recomendada

- Guion píldora formativa Thoth nº 37, ¿Cómo funciona el algoritmo de exponenciación rápida?, Jorge Ramió, 2016
 - <https://www.criptored.es/thoth/material/texto/pildora037.pdf>
- Coursera Square and Multiply
 - <https://es.coursera.org/lecture/mathematical-foundations-cryptography/square-and-multiply-ty62K>
- Modular exponentiation by repeated squaring
 - <https://mathlesstraveled.com/2018/08/18/modular-exponentiation-by-repeated-squaring/>

Class4crypt c4c2.6

Módulo 2. Matemáticas discretas en la criptografía

Lección 2.6. Raíces primitivas en un primo p

2.6.1. Concepto de raíz primitiva

2.6.2. Comprobación de la existencia de raíces primitivas

2.6.3. Búsqueda de raíces primitivas

2.6.4. Tasa de restos que son raíces primitivas

2.6.5. La importancia de los primos seguros

2.6.6. Uso de las raíces primitivas en la criptografía

Class4crypt c4c2.6 Raíces primitivas en un primo p
<https://www.youtube.com/watch?v=dl6NrBH0z9g>




Raíces primitivas o generadores en primos

- Se denomina raíz primitiva α de un primo p al resto x que cumple:
$$\alpha^{x_i} \bmod p = y_i = \text{CRR} \text{ (con } 0 \leq x_i \leq p-1\text{)}$$
- Genera todos los elementos del primo excepto el 0, es decir, el Conjunto Reducido de Restos CRR (1, 2, 3, 4, ... $p-3$, $p-2$, $p-1$)
- Por ello se le conoce también como generador
- Si α es una raíz primitiva entonces se cumple que:
$$\alpha^0 \bmod p = y_0 = 1 \qquad \alpha^{p-1} \bmod p = y_{p-1} = 1$$



Y la operación $\alpha^{x_i} \bmod p = y_{x_i}$ (con $x_i = 1, 2, 3, \dots, p-4, p-3, p-2$) entrega valores diferentes comprendidos entre 2 y $p-1$

Ejemplo: restos raíces y no raíces en $p = 7$

- $\alpha = 2$ no es una raíz primitiva del primo $p = 7$

- $2^0 \bmod 7 = 1$ 
- $2^1 \bmod 7 = 2$
- $2^2 \bmod 7 = 4$
- $2^3 \bmod 7 = 1$ 
- $2^4 \bmod 7 = 2$
- $2^5 \bmod 7 = 4$
- $2^6 \bmod 7 = 1$ 

- $\alpha = 3$ sí es una raíz primitiva del primo $p = 7$

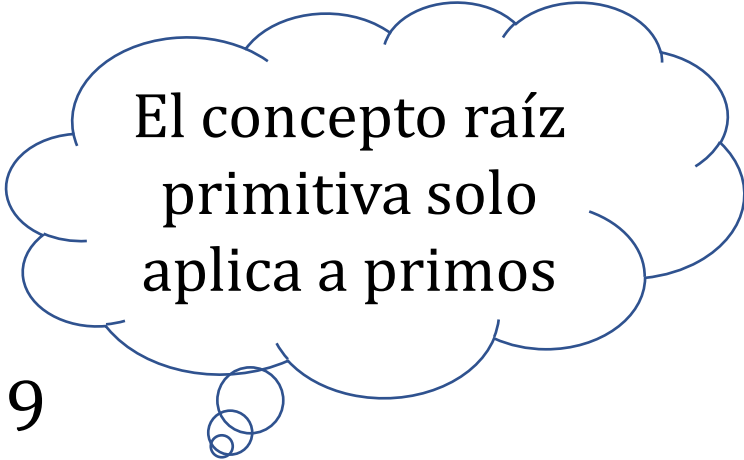
- $3^0 \bmod 7 = 1$ 
- $3^1 \bmod 7 = 3$
- $3^2 \bmod 7 = 2$
- $3^3 \bmod 7 = 6$
- $3^4 \bmod 7 = 4$
- $3^5 \bmod 7 = 5$
- $3^6 \bmod 7 = 1$ 

¿Y si el módulo no fuese primo? (1/2)

- Supongamos $n = 3 \cdot 5 = 15$
- Los restos serían $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$
- Para $\alpha = 2, 3, 4, \dots, 13, 14$, hacemos $\alpha^{x_i} \bmod n = y_i$ (con $0 \leq x_i \leq n-1$)
 - $2^{x_i} \bmod 15 = 1, 2, 4, 8, 1, 2, 4, \dots, 2, 4$
 - $3^{x_i} \bmod 15 = 1, 3, 9, 12, 6, 3, 9, 12, \dots, 3, 9$
 - $4^{x_i} \bmod 15 = 1, 4, 1, 4, 1, \dots, 4, 1$
 - $5^{x_i} \bmod 15 = 1, 5, 10, 5, 10, 5, \dots, 5, 10$
 - $6^{x_i} \bmod 15 = 1, 6, 6, 6, 6, \dots, 6, 6$
 - $7^{x_i} \bmod 15 = 1, 7, 4, 13, 1, 7, 4, \dots, 7, 4$

¿Y si el módulo no fuese primo? (2/2)

- $8^x \bmod 15 = 1, 8, 4, 2, 1, 8, 4, \dots 8, 4$
- $9^x \bmod 15 = 1, 9, 6, 9, 6, 9, \dots 9, 6$
- $10^x \bmod 15 = 1, 10, 10, 10, 10, \dots 10, 10$
- $11^x \bmod 15 = 1, 11, 1, 11, 1, \dots 11, 1$
- $12^x \bmod 15 = 1, 12, 9, 3, 6, 12, 9, 3, \dots 12, 9$
- $13^x \bmod 15 = 1, 13, 4, 7, 1, 13, 4, \dots 13, 4$
- $14^x \bmod 15 = 1, 14, 1, 14, 1, \dots 14, 1$
- Nunca se han obtenido o generado todos los restos de $n = 15$
- Han aparecido unos “anillos”. Esto lo estudiaremos en RSA



El concepto raíz primitiva solo aplica a primos

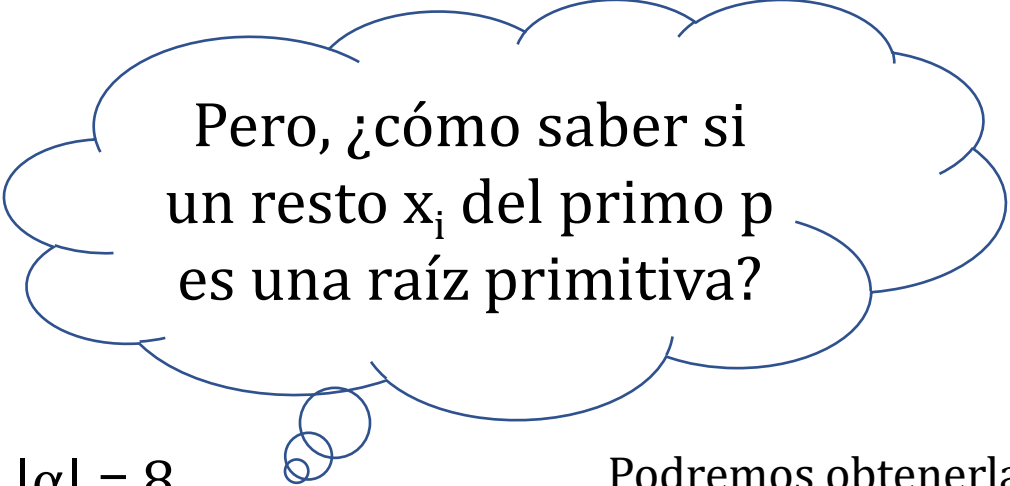
Números candidatos a tener o a ser raíces

- El primo $p = 2$ no tendrá raíces primitivas porque sus restos son 0 y 1 y ambos valores no generan nada puesto que $0^0 \bmod 2 = 0$ y $0^1 \bmod 2 = 0$ (no se obtiene el resto 1) y $1^0 \bmod 2 = 1$ y $1^1 \bmod 2 = 1$ (no se obtiene el resto 0)
- Como $\alpha^{x_i} \bmod p = y_i = \text{CRR}$ (con $0 \leq x_i \leq p-1$) y α son restos de p :
 - El resto 0 nunca será una raíz α de p porque $0^{x_i} \bmod p = 0 \ \forall \ x_i$
 - El resto 1 nunca será una raíz α de p porque $1^{x_i} \bmod p = 1 \ \forall \ x_i$
 - Además, el resto $p-1$ tampoco será nunca una raíz α de p
 - Los demás restos de p , desde 2 hasta $p-2$, sí pueden ser raíces y la cantidad de raíces y valores dependerá del primo en cuestión

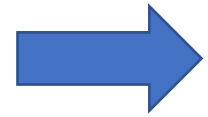
Raíces primitivas de los primeros primos

Raíces calculadas con el programa SAMCript ($|\alpha|$ es la cantidad de raíces)

- $3 = (2) \quad |\alpha| = 1$
- $5 = (2, 3) \quad |\alpha| = 2$
- $7 = (3, 5) \quad |\alpha| = 2$
- $11 = (2, 6, 7, 8) \quad |\alpha| = 4$
- $13 = (2, 6, 7, 11) \quad |\alpha| = 4$
- $17 = (3, 5, 6, 7, 10, 11, 12, 14) \quad |\alpha| = 8$
- $19 = (2, 3, 10, 13, 14, 15) \quad |\alpha| = 6$
- $23 = (5, 7, 10, 11, 14, 15, 17, 19, 20, 21) \quad |\alpha| = 10$
- $29 = (2, 3, 8, 10, 11, 14, 15, 18, 19, 21, 26, 27) \quad |\alpha| = 12$



Pero, ¿cómo saber si
un resto x_i del primo p
es una raíz primitiva?



Podremos obtenerlas con el software

SAMCript

Buscando raíces primitivas en p

- Existen muchos números dentro del CRR que son raíces del primo p , pero su búsqueda no es algo fácil. Necesitamos tener un procedimiento que nos permita encontrar esos números, diferente a aplicar la fuerza bruta
- Si conociendo la factorización de $p-1$ obtenemos (q_1, q_2, \dots, q_n) donde q_i son los factores primos de $p-1$, diremos que un número g será una raíz α de p si
 - $\forall q_i \quad g^{(p-1)/q_i} \bmod p \neq 1$
 - Ejemplo 1: para $p = 13$, como $p-1 = 12 = 2^2 \times 3$, entonces los valores de q_i serán $q_1 = 2, q_2 = 3$
 - Ejemplo 2: para el primo $p = 181$, como $p-1 = 180 = 2^2 \times 3^2 \times 5$, entonces los valores de q_i serán $q_1 = 2, q_2 = 3, q_3 = 5$
- Si alguno de esos resultados es igual a 1, ese valor de g no será una raíz

Raíces primitivas en el primo $p = 13$ (1/3)

Si se cumple $g^{(p-1)/q_i} \bmod p \neq 1 \quad \forall q_i$ entonces g será una raíz de p
Sea $p = 13$, entonces $p-1 = 12 = 2^2 * 3$ y $q_1 = 2$ y $q_2 = 3$

Generadores en \mathbb{Z}_{13}

2

$$2^{(13-1)/2} \bmod 13 = 2^6 \bmod 13 = 12$$

$$2^{(13-1)/3} \bmod 13 = 2^4 \bmod 13 = 3 \quad \text{El resto 2 es una raíz primitiva} \quad \uparrow$$

$$3^{(13-1)/2} \bmod 13 = 3^6 \bmod 13 = 1$$

$$3^{(13-1)/3} \bmod 13 = 3^4 \bmod 13 = 3 \quad \text{El resto 3 no es una raíz primitiva}$$

$$4^{(13-1)/2} \bmod 13 = 4^6 \bmod 13 = 1$$

$$4^{(13-1)/3} \bmod 13 = 4^4 \bmod 13 = 9 \quad \text{El resto 4 no es una raíz primitiva}$$

Raíces primitivas en el primo $p = 13$ (2/3)

Generadores en \mathbb{Z}_{13}

2 6 7

$$5^{(13-1)/2} \bmod 13 = 5^6 \bmod 13 = 12$$

$$5^{(13-1)/3} \bmod 13 = 5^4 \bmod 13 = \mathbf{1} \quad \text{El resto 5 no es una raíz primitiva}$$

$$6^{(13-1)/2} \bmod 13 = 6^6 \bmod 13 = 12$$

$$6^{(13-1)/3} \bmod 13 = 6^4 \bmod 13 = 9 \quad \text{El resto 6 es una raíz primitiva} \quad \uparrow$$

$$7^{(13-1)/2} \bmod 13 = 7^6 \bmod 13 = 12$$

$$7^{(13-1)/3} \bmod 13 = 7^4 \bmod 13 = 9 \quad \text{El resto 7 es una raíz primitiva} \quad \uparrow$$

$$8^{(13-1)/2} \bmod 13 = 8^6 \bmod 13 = 12$$

$$8^{(13-1)/3} \bmod 13 = 8^4 \bmod 13 = \mathbf{1} \quad \text{El resto 8 no es una raíz primitiva}$$

Raíces primitivas en el primo $p = 13$ (3/3)

Generadores en \mathbb{Z}_{13}

2 6 7 11

$$9^{(13-1)/2} \bmod 13 = 9^6 \bmod 13 = \mathbf{1}$$

$$9^{(13-1)/3} \bmod 13 = 9^4 \bmod 13 = 9 \quad \text{El resto 9 no es una raíz primitiva}$$

$$10^{(13-1)/2} \bmod 13 = 10^6 \bmod 13 = \mathbf{1}$$


$$10^{(13-1)/3} \bmod 13 = 10^4 \bmod 13 = 3 \quad \text{El resto 10 no es una raíz primitiva}$$

$$11^{(13-1)/2} \bmod 13 = 11^6 \bmod 13 = 12$$

$$11^{(13-1)/3} \bmod 13 = 11^4 \bmod 13 = 3 \quad \text{El resto 11 es una raíz primitiva} \quad \uparrow$$

$$\alpha_{13} = (2, 6, 7, 11)$$

¿Cuántas raíces primitivas tiene un primo?

- La tasa τ de raíces primitivas en el primo p será aproximadamente
 - $\tau = \phi(p-1)/(p-1)$
 - Donde $\phi(p-1)$ es el Indicador de Euler de $(p-1)$, es decir la cantidad de restos de $(p-1)$ que no tienen factores en común con $(p-1)$, además del 1
- Por ejemplo, si $p = 41$, entonces $p-1 = 40$ y $\phi(40) = 16$ ($40 = 2^3 \cdot 5$)
 - 16 restos: 1, 3, 7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31, 33, 37, 39
 - $\tau = \phi(40)/40 = 16/40 = 0,40$ (un 40%)
- Pregunta: ¿obtendremos siempre un porcentaje similar? 

La tasa τ de raíces depende del primo

• Si $p = 3$, $p-1 = 2$	$\tau = \phi(2)/2 = 1/2 = 0,50$	50%
• Si $p = 5$, $p-1 = 4$	$\tau = \phi(4)/4 = 2/4 = 0,50$	50%
• Si $p = 7$, $p-1 = 6$	$\tau = \phi(6)/6 = 2/6 = 0,33$	33%
• Si $p = 11$, $p-1 = 10$	$\tau = \phi(10)/10 = 4/10 = 0,40$	40%
• Si $p = 13$, $p-1 = 12$	$\tau = \phi(12)/12 = 4/12 = 0,33$	33%
• Si $p = 17$, $p-1 = 16$	$\tau = \phi(16)/16 = 8/16 = 0,50$	50%
• Si $p = 19$, $p-1 = 18$	$\tau = \phi(18)/18 = 6/18 = 0,33$	33%
• Si $p = 23$, $p-1 = 22$	$\tau = \phi(22)/22 = 10/22 = 0,45$	45%
• Si $p = 29$, $p-1 = 28$	$\tau = \phi(28)/28 = 12/28 = 0,43$	43%
• Si $p = 31$, $p-1 = 30$	$\tau = \phi(30)/30 = 8/30 = 0,27$	27%

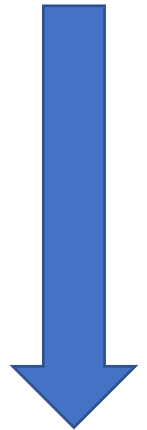
El porcentaje τ de restos que serán raíces primitivas se encontrará entre el 25% y el % 50%

¿Con qué tipo de primos obtenemos una tasa τ mayor?



Raíces primitivas en primos seguros

- Primo seguro $p = 2q + 1$, siendo q un primo (interesante en criptografía asimétrica)
- 5, 7, 11, 23, 47, 59, 83, 107, 167, 179, 227, 263, 347, 359, 383, 467, 479, ...
- Si $p = 47$, $p-1 = 46$ $\tau = \phi(46)/46 = 22/46 = 0,4783$ 47,83%
- Si $p = 59$, $p-1 = 58$ $\tau = \phi(58)/58 = 28/58 = 0,4828$ 48,28%
- Si $p = 83$, $p-1 = 82$ $\tau = \phi(82)/82 = 40/82 = 0,4878$ 48,78%
- Si $p = 107$, $p-1 = 106$ $\tau = \phi(106)/106 = 52/106 = 0,4906$ 49,06%
- Si $p = 167$, $p-1 = 166$ $\tau = \phi(166)/166 = 82/166 = 0,4940$ 49,40%
- Si $p = 179$, $p-1 = 178$ $\tau = \phi(178)/178 = 88/178 = 0,4944$ 49,44%
- Si $p = 227$, $p-1 = 226$ $\tau = \phi(226)/226 = 112/226 = 0,4956$ 49,56%
- $p = 1.048.343$, $p-1 = 1.048.342$ $\tau = \phi(p-1)/p-1 = 524.170/1.048.342$ 49.9999046%



Primos seguros entregan una mayor tasa τ

- A medida que el primo seguro es mayor, la tasa τ se va acercando al valor del 50%
- Comprobación. Sabemos que $\tau = \phi(p-1)/(p-1)$
- Si $p = 2q + 1$, entonces $p-1 = 2q$
- Por lo tanto $\tau = \phi(p-1)/(p-1) = \phi(2q)/2q$
- Pero $\phi(2q) = \phi(2) * \phi(q) = 1 * \phi(q) = q-1$
- Luego $\tau = (q-1)/2q$
- Si q es muy grande, entonces $q-1 \approx q$ y por tanto $\tau \approx \frac{1}{2}$ (50%)

Distribución de raíces en primos seguros


- Ejemplo: distribución de las 52 raíces primitivas de $p = 107$
 - $\{2, 5, 6, 7, 8, 15, 17, 18, 20, 21, 22, 24, 26, 28, 31, 32, 38, 43, 45, 46, 50, 51, 54, 55, 58, 59, 60, 63, 65, 66, 67, 68, 70, 71, 72, 73, 74, 77, 78, 80, 82, 84, 88, 91, 93, 94, 95, 96, 97, 98, 103, 104\}$
 - Se asemeja a una distribución uniforme, hay largas cadenas de números seguidos y consecutivos $\{5, 6, 7, 8\}$, $\{20, 21, 22\}$, $\{58, 59, 60\}$, $\{65, 66, 67, 68\}$, $\{70, 71, 72, 73, 74\}$, $\{93, 94, 95, 96, 97, 98\}$ y es bastante frecuente observar una separación entre números raíces igual a 2 o 3
- Por tanto, si en la ecuación $g^{(p-1)/q_i} \bmod p \neq 1$ para saber si g es una raíz primitiva de p , elegimos un valor g que no resulta ser una raíz primitiva, la posibilidad de que $g+1$ o $g+2$ sí lo sea, será muy alta


Uso de raíces primitivas en la criptografía

- La utilidad de este concepto en criptografía lo veremos cuando se estudien los sistemas de clave pública y, en particular, el protocolo de intercambio de claves de Diffie Hellman
- También se recurrirá a esto cuando estudiemos la firma digital según Elgamal y el estándar de firma digital DSA Digital Signature Algorithm
- Pregunta. ¿Será fácil encontrar las primeras raíces primitivas de un primo p muy grande, por ejemplo de 2.048 bits? No, porque habrá que factorizar el valor $p-1$, que será un trabajo muy costoso
- ¿Solución? Usar como raíz primitiva un número pequeño (por ejemplo 2 o 5), aunque no lo sea, sin comprobarlo, tal y como lo hace OpenSSL **¿inseguro?** ➡
- Como los números son tan grandes, en la práctica el sistema no será inseguro

Importancia de las raíces primitivas

- En criptografía de clave pública, por ejemplo intercambio de clave de Diffie y Hellman, el usuario calcula su clave pública **y** usando la siguiente ecuación:
 $y = \alpha^x \bmod p$, siendo p un primo, α una raíz primitiva de p y x su clave privada

- Sea $x = 18$, $p = 31$, $\alpha = 3$ (raíz)
- $y = 3^{18} \bmod 31 = 4$ 
- Como 3 es una raíz primitiva de 31, el único valor privado que entrega un valor público 4 será el número 18, porque $y = 3^{x_i} \bmod 31$ entrega todos sus resultados diferentes

- Sea $x = 18$, $p = 31$, $\alpha = 2$ (no raíz)
- $y = 2^{18} \bmod 31 = 8$ 
- Como 2 no es una raíz primitiva de 31, habrá más de un número x_i que la ecuación $y = 2^{x_i} \bmod 31$ entregue como resultado el valor público 8
- Por ejemplo $x_i = 3, 8, 13, 18, 23$ y 28

Conclusiones de la Lección 2.6

- Los restos α son raíces primitivas en un primo p si cumplen con la condición de que $\alpha^{x_i} \bmod p = y_i$ entrega números distintos para $0 \leq x_i \leq p-1$
- En este caso de raíz primitiva, para un valor y_i existirá un único número x_i que cumpla con la ecuación anterior (muy interesante en criptografía asimétrica)
- Si no se usa una raíz primitiva α sino otro resto g , entonces habrá al menos dos valores de x_i que $g^{x_i} \bmod p = y_i$ entregue el mismo valor y_i
- Si x es una clave privada que genera una clave pública $y = \alpha^x \bmod p$, entonces encontrar qué exponente x le corresponde al número y , significa resolver el problema del logaritmo discreto PLD, de muy difícil solución si p es grande
- La tasa de raíces primitivas τ se encuentra entre el 25% y el 50%
- Si p es un primo seguro, la tasa de raíces primitivas τ acerca al 50%

Lectura recomendada

- Euler's Totient Calculator
 - <http://www.javascripter.net/math/calculators/eulertotientfunction.htm>
- List of prime numbers, List of safe primes / Wikipedia
 - https://en.wikipedia.org/wiki/List_of_prime_numbers
 - <https://prime-numbers.info/list/safe-primes-page-4>
- Primitive root modulo n / Wikipedia
 - https://en.wikipedia.org/wiki/Primitive_root_modulo_n
- SAMCrypt: Software de Aritmética Modular para Criptografía, María Nieto Díaz, dirección Jorge Ramió, 2018
 - https://www.criptored.es/software/sw_m001t.htm