PRIMERA PARTE

ALGORITMO PARA EL CÁLCULO DE (P, q)

PARA UN PROBLEMA DE LOGARITMO DISCRETO

1) SE CALCULA UN PRIMO GRANDE q ($q > 2^{1000}$)

2) PARA $K = 1, 2, \ldots$

SI $P := 2qK + 1$ ES PRIMO, NOS DETENEMOS Y FIJAMOS P

NOTA: SI EL PROCESO ANTERIOR TERMINA PRONTO $P - 1 = 2qK$ TIENE FACTORIZACIÓN FÁCIL (CONOCEMOS q)

3) SE FACTORIZA

$$P - 1 = P_1^{s_1} P_2^{s_2} \cdots P_t^{s_t}$$

, CON $P_1 = q$ $P_2 = 2$

$K = P_1^{s_1 - 1} P_2^{s_2 - 1} P_3^{s_3} \cdots P_t^{s_t}$

4) BUSCAMOS AL AZAR (I.E, VAMOS PROBANDO) UN ENTERO g, $1 < g < p-1$ TAL QUE

$$g^{\frac{p-1}{p_i}} \neq 1 \mod p \text{, PARA } i = 1, \dots, t$$

5) ENTONCES DEVOLVEMOS (OUTPUT) EL PAR (p, g) PUESTO QUE g ES UN GENERADOR MULTIPLICATIVO MODULO p.

PORQUE:

Th: SEA $n \in \mathbb{N}$. SI $g^n = 1$ Y $g^{n/p} \neq 1$ PARA TODO DIVISOR PRIMO p DE n ENTONCES EL ORDEN DE g ES n.

DEM: EL ORDEN DE UN ELEMENTO DEBE DIVIDIR AL ORDEN (TAMAÑO) DEL GRUPO

¿CÓMO DE RÁPIDO ES EL ANTERIOR ALGORITMO?

PASO 2): ¿VA A PARAR PRONTO?

SIEMPRE QUE EN

$1+2q,\ 1+2(2q),\ 1+3(2q),\ \ldots$

$a \quad a \quad d \quad d$

HAYA MUCHOS PRIMOS, PARA ENCONTRAR UNO PRONTO

Th DIRICHLET:

SEAN a, d ENTEROS POSITIVOS CON $\mathrm{mcd}(a,d)=1$

LA PROGRESIÓN $a,\ a+d,\ a+2d,\ \ldots$

CONTIENE INFINITOS PRIMOS

DE HECHO, PARA x SUFICIENTEMENTE GRANDE EL NÚMERO DE PRIMOS DE LA PROGRESIÓN MENORES O IGUALES QUE x ES APROXIMADAMENTE

$$\frac{x}{\varphi(d)\ln x}$$

SOBRE EL PASO 4)

Th: HAY $\varphi(p-1)$ ELEMENTOS QUE SON GENERADORES DE $(\mathbb{Z}_p)^*$

NO QUERÍAMOS $p-1$ CON FACTORES PEQUEÑOS $\Rightarrow$

$\Rightarrow \varphi(p-1)$ SEA GRANDE $\Rightarrow$ HAY MUCHOS g's QUE PODEMOS ENCONTRAR

UNA ALTERNATIVA

1') SE CALCULA UN PRIMO GRANDE q

2') SI $p := 2q+1$ ES PRIMO. SEGUIMOS, SI NO VOLVEMOS A 1') PARA CALCULAR OTRO PRIMO q

1')+2') ⎧
- q PRIMO GRANDE
- WHILE $p=2q+1$ NO SEA PRIMO ⎨ q = PRIMO GRANDE AL AZAR ⎩

← SIN "GO TO"

← ASEGURA $k=1$

DE ESTA FORMA $p-1 = 2 \cdot q$ (3) ES "GRATIS")

(2 = PRIMO, q = PRIMO)

Y EN 4) SÓLO HAY QUE COMPROBAR QUE

$$g^2 \not\equiv 1 \bmod p \quad \text{Y} \quad g^q \not\equiv 1 \bmod p$$

$\varphi(p-1) = \varphi(q) = q-1$

VENTAJAS DE CALCULAR (p, g) DE ESTA MANERA

HAY MUCHOS GENERADORES

1) ASÍ g ES UN GENERADOR DEL GRUPO MULTIPLICATIVO $(\mathbb{Z}_p)^*$ (TIENE EL ORDEN MAYOR POSIBLE)

2) LA PERSONA QUE ENVÍA EL MENSAJE PUEDE COMPROBAR FÁCILMENTE QUE EL RECEPTOR HA ESCOGIDO (p, g) DE FORMA ADECUADA Y QUE SU MENSAJE NO VA A SER INTERCEPTADO Y DESCIFRADO POR UN CRIPTOANALISTA. POR EJEMPLO, SI EL ORDEN DE g NO ES LO SUFICIENTEMENTE GRANDE

PARA UNA CLAVE PÚBLICA (p, g, Δ) GENERADA CON EL ANTERIOR ALGORITMO

EMISOR MIRA QUE

- p SEA PRIMO (TEST PROBABILISTICO)
- g SEA UN GENERADOR DE $\mathbb{Z}_p^*$
  - $g^2 \not\equiv 1 \mod p$
  - $g^q \not\equiv 1 \mod p$, DONDE $q = \frac{p-1}{2}$

ALGORITMO DE LOS CUADRADOS REPETIDOS

TANTO PARA EL CAMAL COMO PARA RSA NECESITAMOS EXPONENCIAR NÚMEROS GRANDES MÓDULO UN ENTERO. VEAMOS QUE ESTO PUEDE HACERSE DE FORMA EFICIENTE:

EJ: $12^{11} \bmod 21$

$a^{b+c} = a^b a^c$

11 BINARIO: 1011, ES DECIR

$$11 = 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 2^3 + 2 + 1$$

$$12^{11} = 12^{2^3+2+1} = 12^{2^3} \cdot 12^2 \cdot 12^1$$

$12 \bmod 21 = [12]$ ← HACER MOD , con $0 \leq [12] < 20$

$12^{11} \bmod 21 = [12^{11}] = [[12^{2^3}] \cdot [12^2] \cdot [12]]$

$[12^1] = 12$

$[12^2] = [144] = 18$

$[12^{2^2}] = [(12^2)^2] = [[12^2] \cdot [12^2]] = [18 \cdot 18] \overset{*}{\equiv} 9$

$[12^{2^3}] = [12^{2^2 \cdot 2}] \overset{2^2 \cdot 2 = 2^{2+1} = 2^3}{=} [(12^{2^2})^2] = [[12^{2^2}][12^{2^2}]]$

$= [9 \cdot 9] = 18$

$[12^{2^4}] = [18 \cdot 18] = 9$

$$[12^{11}] = [[12^{2^3}][12^{2^1}][12]] =$$

$$= [18 \circ 18 \cdot 12] = [[18 \cdot 18] \cdot [12]]$$

$$\overset{(*)}{=} [9 \cdot 12] = 3$$

$(a^b)^c = a^{b \cdot c}$

$a^b \cdot a^c = a^{b+c}$

TRUCO: $[a^{2^n}] = [a^{2^{n-1} \cdot 2}] = [(a^{2^{n-1}})^2]$

$$= [[a^{2^{n-1}}][a^{2^{n-1}}]]$$

$[a^1]$

$[a^2]$

$[a^{2^2}]$

$[a^{2^3}]$

$[a^{2^4}] \ldots$

SE TERMINA REPITIENDO

NUNCA HACEMOS UNA MULTIPLICACIÓN MAYOR QUE $(p-1)^2$

ALGORITMO EXPONENCIACIÓN MODULAR

INPUT (b ENTERO, n ENTERO, m ENTERO)

OUTPUT $b^n \bmod m$

$a_{k-1} \cdots a_1 a_0 :=$ n "EN BASE 2"

$x := 1;$

Power $:= b \bmod m;$

FOR $i = 0$ TO $k-1$ DO

IF $a_i = 1$ THEN

$x := x \cdot$ Power $\bmod m;$

Power $:=$ Power $\cdot$ Power $\bmod m;$

RETURN $x;$

Power_mod EN SAGE

IDEA KARATSUBA

$$(ax+b)\cdot(cx+d) = \underset{1}{ac}x^2 + (\underset{2}{ad} + \underset{1}{bc})x + \underset{4}{bd}$$

- 4 MULTIPLICACIONES
- 1 SUMA

MEJOR:

CALCULAMOS $\underset{1}{ac}$, $\underset{2}{bd}$ Y

$$(\underset{1}{a}+\underset{3}{b})(\underset{2}{c}+d) - \underset{3}{ac} - \underset{4}{bd} = ad+bc$$

(ac ya lo tenemos; bd ya lo tenemos)

- 3 MULTIPLICACIONES
- 4 SUMAS (O RESTAS)

MULTIPLICAR ES MÁS COSTOSO QUE SUMAR

ALGORITMO KARATSUBA

MULTIPLICACIÓN MODULAR

$$a = (a_{2n-1}\ a_{2n-2}\ \cdots\ a_0)_2$$

$$b = (b_{2n-1}\ b_{2n-2}\ \cdots\ b_0)_2$$

¿$a \cdot b$?

SEAN

$$a = 2^n A_1 + A_0 \quad , \quad b = 2^n B_1 + B_0$$

$A_i < 2^n$

$B_i < 2^n$

- $A_1 = (a_{2n-1}\ \cdots\ a_{n+1}\ a_n)_2$
- $A_0 = (a_{n-1}\ \cdots\ a_1\ a_0)_2$
- $B_1 = (b_{2n-1}\ \cdots\ b_{n+1}\ b_n)_2$
- $B_0 = (b_{n-1}\ \cdots\ b_1\ b_0)_2$

$$ab = A_1 B_1 2^{2n} + (A_1 B_0 + A_0 B_1) 2^n + A_0 B_0$$

4 MULTIPLICACIONES
3 SUMAS

$$= A_1 B_1 2^{2n} + (A_1 B_1 + A_0 B_0 - (A_1 - A_0)(B_1 - B_0)) 2^n$$

CESAR CRATIS

$$+ A_0 B_0$$

$$A_1 B_1 + A_0 B_0 - (A_1 B_1 - A_1 B_0 - A_0 B_1 + A_0 B_0)$$

$$A_1 B_1 + A_0 B_0 - A_1 B_1 + A_1 B_0 + A_0 B_1 - A_0 B_0$$

$$A_1 B_0 + A_0 B_1$$

$\Rightarrow$ (azul) = (verde)

3 MULTIPLICACIONES
4 SUMAS

$$ab = (2^{2n} + 2^n) A_1 B_1 + 2^n (A_1 - A_0)(B_0 - B_1) +$$

① ②

$$+ (2^n + 1) A_0 B_0$$

③

Y VAMOS LLAMANDO RECURSIVAMENTE AL ALGORITMO 3 VECES

KARATSUBA $\Theta(n^{\log 3})$

HABITUAL: $\Theta(n^2)$

EJ: KARATSUBA PARA MULTIPLICAR 5 Y 7

$a = 5 = (0101)_2$

$b = 7 = (0111)_2$

$n = 2 \quad 5 = 2^2 A_1 + A_0$

$7 = 2^2 B_1 + B_0$

$A_1 = (01)_2 \quad A_0 = (01)_2$

$B_1 = (01)_2 \quad B_0 = (11)_2$

$A_0 B_0 = (11)_2$

$A_1 B_1 = (01)_2$

$(A_1 - A_0)(B_1 - B_0) = ((01)_2 - (01))_2 ((01)_2 - (11)_2) = (00)_2$

AQUÍ MULTIPLICO DIRECTAMENTE PERO ESTAS 3 OPERACIONES SE HARÍAN DE FORMA RECURSIVA

$$ab = (2^{2n} + 2^n) A_1 B_1 + 2^n (A_1 - A_0)(B_0 - B_1) + (2^n + 1) A_0 B_0$$

$$= (2^4 + 2^2)(01)_2 + 2^2 (00)_2 + (2^2 + 1)(11)_2 =$$

$$= (10100)_2 (01)_2 + (101)(11)_2$$

$$= (10100)_2 + (1111)_2 = 35$$

(20) (15)

EL ALGORITMO ES RECURSIVO, PARA HACER UNA MULTIPLICACIÓN COMO $A_0 B_0 = (01)_2 \cdot (11)_2$ SE LLAMARÍA AL ALGORITMO DE NUEVO CON $A_0' = 1$ $A_1' = 1$ $B_0' = 1$ $B_1' = 1$