

# Teoría de Números y Criptografía

F. J. Lobillo

2021/2022



# **Parte II**

## **Criptografía y Curvas Elípticas**



# Índice general

<b>II</b>	<b>Criptografía y Curvas Elípticas</b>	<b>2</b>
<b>1.</b>	<b>Complejidad algorítmica</b>	<b>6</b>
1.1.	Introducción . . . . .	6
	Ejercicios de Complejidad algorítmica . . . . .	11
<b>2.</b>	<b>Criptografía simétrica</b>	<b>13</b>
2.1.	Cifrado y secreto . . . . .	13
2.2.	Objetivos de la criptografía . . . . .	14
2.3.	Ataques . . . . .	15
2.4.	Seguridad probable . . . . .	16
2.5.	Criptografía simétrica . . . . .	17
2.6.	Cifrados de flujo . . . . .	18
2.7.	Cifrados de bloque . . . . .	20
2.7.1.	Modos de operación . . . . .	20
2.8.	Apéndice: Sistemas de numeración . . . . .	23
	Ejercicios de Criptosistemas simétricos . . . . .	24
	Ejercicios de evaluación de Criptosistemas simétricos . . . . .	29

<b>3. RSA</b>	<b>30</b>
3.1. Función unidireccional . . . . .	30
3.2. Descripción de RSA . . . . .	38
3.3. Ataques . . . . .	42
Ejercicios de RSA . . . . .	58
Ejercicios de evaluación del Criptosistema RSA . . . . .	59
<b>4. Logaritmo discreto</b>	<b>60</b>
4.1. Problema del logaritmo discreto . . . . .	60
4.1.1. Paso de bebé – Paso de gigante. . . . .	61
4.1.2. El algoritmo de Silver-Pohlig-Hellman . . . . .	64
4.1.3. Cálculo de índices en cuerpos primos . . . . .	67
4.1.4. Cálculo de índices en cuerpos finitos . . . . .	70
4.2. Protocolo de Diffie-Hellman . . . . .	75
4.3. Criptosistema de ElGamal . . . . .	77
4.4. Digital Signature Algorithm . . . . .	79
Ejercicios de logaritmo discreto . . . . .	83
Ejercicios de evaluación de logaritmo discreto . . . . .	85
<b>5. Curvas elípticas</b>	<b>86</b>
5.1. Concepto de curva elíptica. . . . .	86
5.2. Curvas elípticas proyectivas . . . . .	93
5.3. Aritmética de una curva elíptica . . . . .	95
5.4. Teoremas de Hasse y Rück . . . . .	116
5.5. Orden de puntos y curvas . . . . .	117
5.5.1. Puntos de la curva . . . . .	117
5.5.2. Orden de puntos . . . . .	131
5.5.3. Cardinal de la curva . . . . .	134

Curvas elípticas . . . . .	137
Ejercicios de evaluación de curvas elípticas . . . . .	140
<b>6. Criptosistemas basados en curvas elípticas</b>	<b>141</b>
6.1. Aritmética en característica $p > 3$ . . . . .	141
6.2. Aritmética en característica 2 . . . . .	142
6.3. Complejidad de la aritmética en EC . . . . .	144
6.4. Parámetros para uso criptográfico . . . . .	145
6.5. Protocolo ECDH . . . . .	147
6.6. Criptosistema ElGamal en EC . . . . .	148
6.7. ECDSA . . . . .	149
6.8. Codificación de mensajes . . . . .	151
6.9. Criptosistema de Menezes-Vanstone . . . . .	152
6.10. Curvas en OpenSSL . . . . .	153
Curvas elípticas . . . . .	159
Ejercicios de evaluación de criptosistemas basados en curvas elípticas . . . . .	160



# Criptosistemas basados en el logaritmo discreto

4.1

## Problema del logaritmo discreto

Sea  $G$  un grupo y  $b \in G$  un elemento de orden  $n$ . Dado  $h \in \langle b \rangle$ , el problema del logaritmo discreto consiste en encontrar  $0 \leq m \leq n - 1$  tal que  $h = b^m$ . En este caso decimos

$$m = \log_b(h) = \text{idx}_b(h).$$

Si empleamos notación aditiva para la operación en  $G$ , el problema se traduce en encontrar  $0 \leq m \leq n - 1$  tal que  $h = mb$ .

La Proposición 1.8 establece el coste de calcular potencias en el grupo multiplicativo  $\mathcal{U}(\mathbb{Z}_n)$ , y este resultado puede extenderse a cualquier grupo.

**Proposición 4.1.** *El tiempo empleado en calcular  $g^m$  es*

$$\mathcal{O}(\max\{M(G)(\log m), (\log m)^2\}),$$

donde  $M(G)$  es el tiempo empleado en realizar la multiplicación en  $G$ .

*Demostración.* Análogo a la demostración de la Proposición 1.8, usando cuadrados iterados.  $\square$

Por tanto, si hay una forma eficiente de calcular la operación en  $G$ , podremos eficientemente calcular potencias. Sin embargo no se conocen algoritmos eficientes para calcular logaritmos discretos en general. Presentamos algunos de los clásicos, aunque ninguno tiene complejidad polinomial.

#### 4.1.1 Paso de bebé – Paso de gigante.

Este algoritmo se conoce a veces como **algoritmo de Shanks**, en honor a Daniel Shanks, el primero que lo publicó, aunque algunos autores aseguran que era conocido previamente.

**Teorema 4.2.** *El Algoritmo 3 calcula el logaritmo discreto, si existe.*

*Demostración.* Observemos que  $h_i = hb^{-if}$ . Por tanto, si  $h_i = b^j$ , tenemos que

$$h = hb^{-if}b^{if} = b^jb^{if} = b^{j+if},$$

de donde  $j + if = \log_b(h)$ . Por tanto, el algoritmo da la salida correcta si existe el logaritmo discreto.

Por otra parte, si  $h = b^m$ , dividimos  $m$  entre  $f$ , obteniendo  $m = if + j$  donde  $0 \leq j < f$ . Dado que  $m < n$ , necesariamente  $i < f$ , por lo que

$$h_i = hb^{-if} = b^mb^{-if} = b^{if+j}b^{-if} = b^j,$$

es decir, si existe el logaritmo discreto el algoritmo debe encontrarlo.  $\square$

**Algorithm 3** Baby step – Giant step**Input:**  $b \in G$  de orden  $n$ , y  $h \in \langle b \rangle$ **Output:**  $\log_b h$ 

- 1:  $f \leftarrow \lceil \sqrt{n} \rceil$
- 2:  $\text{table} \leftarrow []$
- 3: **for**  $0 \leq i \leq f - 1$  **do**
- 4:    $\text{table} \leftarrow \text{table} + [(i, b^i)]$
- {La tabla construida es

$$\text{table} = \begin{bmatrix} 0 & 1 & \dots & f-1 \\ b^0 & b^1 & \dots & b^{f-1} \end{bmatrix}$$

}

- 5: Calcula  $b^{-f} = b^{n-f}$
- 6:  $h_0 \leftarrow h$
- 7: **for**  $0 \leq i \leq f - 1$  **do**
- 8:   **if**  $\exists (j, h_i) \in \text{table}$  **then**
- 9:     **return**  $j + if$
- 10:   **else**
- 11:      $h_{i+1} \leftarrow h_i b^{-f}$
- 12: **return** Error, no existe el logaritmo.





Este algoritmo nos fuerza a realizar  $f = \lceil \sqrt{n} \rceil$  multiplicaciones para calcular  $\text{table}$ ,  $3 \log_2(n - f)$  multiplicaciones para calcular  $b^{-f}$ , y un máximo de  $f$  multiplicaciones para calcular los  $h_i$ . Además hay que realizar en media  $\frac{f}{2}$  comparaciones, u ordenar inicialmente  $\text{table}$ . Esto hace que este algoritmo sea impracticable para valores grandes de  $n$ .

**Ejemplo 4.3.** Sea  $p = 251$  y  $b = 6$ , un elemento primitivo módulo  $p$ . Vamos a calcular  $\log_6(20)$  empleando el algoritmo de Shanks. Para ello,

$$f = \lceil \sqrt{p-1} \rceil = 16.$$

Calculamos  $\text{table}$ , y obtenemos

0	1	2	3	4	5	6	7										
1	6	36	216	41	246	221	71										
								8	9	10	11	12	13	14	15		
								175	46	25	150	147	129	21	126		

El siguiente paso consiste en calcular

$$b^{-f} \equiv b^{p-1-f} = 6^{234} \equiv 84 \pmod{251}.$$

Empezando por  $h_0 = 20$ , vamos calculando parejas  $(i, h_i)$  donde  $h_i = h_{i-1} b^{-f} \pmod{p}$  hasta encontrar una en la que  $h_i \in \text{table}$ . Las parejas calculadas son las siguientes

$$(0, 20), (1, 174), (2, 58), (3, 103), (4, 118), (5, 123), (6, 41)$$

y como  $(4, 41) \in \text{table}$ , tenemos que  $\log_6(20) = j + if = 4 + 6 \times 16 = 100$ .

**Algorithm 4** Silver-Pohlig-Hellman**Input:**  $b \in G$  de orden  $n$ , y  $h \in \langle b \rangle$ ,  $n = \prod_{i=1}^r p_i^{e_i}$ **Output:**  $\log_b h$ 1: **for**  $1 \leq i \leq r$  **do**2:   **for**  $0 \leq j \leq p_i - 1$  **do**3:      $r_{i,j} \leftarrow b^{jn/p_i}$     {Estos valores son las raíces  $p_i$ -ésimas de la unidad en  $\langle b \rangle$ }4:   **for**  $0 \leq k \leq e_i - 1$  **do**5:      $y_k \leftarrow h/b^{x_0 + x_1 p_i + \dots + x_{k-1} p_i^{k-1}}$ 6:      $x_k \leftarrow j$  tal que  $y_k^{n/p_i^{k+1}} = r_{i,j}$ 7:    $m_i \leftarrow x_0 + x_1 p_i + \dots + x_{e_i-1} p_i^{e_i-1}$ 8: **return**  $m$  tal que  $m \equiv m_i \pmod{p_i^{e_i}}, 1 \leq i \leq r$ 

## 4.1.2 El algoritmo de Silver-Pohlig-Hellman

**Teorema 4.4.** El Algoritmo 4 calcula el logaritmo discreto.

*Demostración.* Supongamos que  $m = \log_b h$ . Dado que  $n = \prod_{i=1}^r p_i^{e_i}$ , por el Teorema Chino del Resto es suficiente con calcular  $m_i = m \pmod{p_i^{e_i}}, 1 \leq i \leq r$  para conocer  $m$ . Para simplificar la notación, sean  $i \in \{1, \dots, r\}$ ,  $p = p_i$  y  $e = e_i$ . Sea, además  $r_j = b^{jn/p}$  para  $0 \leq j \leq p - 1$ . Dado que  $b^n = 1$ , tenemos que  $\{r_0, \dots, r_{p-1}\}$  son todas las raíces  $p$ -ésimas de la unidad. Supongamos que

$$m \equiv x_0 + x_1 p + \dots + x_{e-1} p^{e-1} \pmod{p^e}$$

con  $0 \leq x_k < p$ . Como  $h = b^m$ , tenemos que

$$h^{n/p} = b^{mn/p} = b^{x_0 n/p} = r_{x_0},$$

por tanto calculamos  $x_0$  encontrando el valor de  $0 \leq j \leq p-1$  tal que  $h^{n/p} = r_j$ . Supongamos que hemos calculado  $x_0, \dots, x_{k-1}$  y sea

$$y_k = h/b^{x_0 + x_1 p + \dots + x_{k-1} p^{k-1}}.$$

Como

$$\begin{aligned} y_k &= b^{n - (x_0 + x_1 p + \dots + x_{k-1} p^{k-1})} \\ &= b^{x_k p^k + \dots + x_{e-1} p^{e-1} + \alpha p^e}, \end{aligned}$$

tenemos que

$$\begin{aligned} y_k^{n/p^{k+1}} &= b^{(x_k p^k + \dots + x_{e-1} p^{e-1} + \alpha p^e) n / p^{k+1}} \\ &= b^{(x_k + \dots + x_{e-1} p^{e-1-k} + \alpha p^{e-k}) n / p} \\ &= b^{x_k n / p} b^{(x_{k+1} + \dots + x_{e-1} p^{e-1-k-1} + \alpha p^{e-k-1}) n} \\ &= b^{x_k n / p} \\ &= r_{x_k}, \end{aligned}$$

por tanto  $x_k$  es el valor  $0 \leq j \leq p-1$  tal que  $y_k^{n/p^{k+1}} = r_j$ . Así calculamos  $x_0, \dots, x_{e-1}$ , lo que termina el teorema.  $\square$

Este algoritmo sólo es útil si podemos factorizar  $n$ , lo cual es posible si los primos que aparecen en la factorización de  $n$  son pequeños. Además, si uno de los factores de  $n$  es grande, el número de raíces a precalcular hace también inaplicable el Algoritmo 4.

**Ejemplo 4.5.** Sea  $p = 397$  y  $G = \mathbb{Z}_p^*$  un grupo cíclico de orden  $p-1 = 396 = 2^2 \cdot 3^2 \cdot 11$ . Un generador de  $G$  es  $b = 5$ . Vamos a calcular  $\log_5(337)$ .

La primera vuelta del algoritmo trabaja con  $p_1 = 2$  y  $e_1 = 2$ . Tenemos que

$$r_{1,0} = 1, r_{1,1} = 396,$$

y la sucesión de coeficientes

$$y_0 = 337, x_0 = 1, y_1 = 385, x_1 = 1,$$

por lo que  $m_1 = 1 + 1 \cdot 2 = 3$ .

En la segunda vuelta,  $p_2 = 3$ ,  $e_2 = 2$ ,

$$r_{2,0} = 1, r_{2,1} = 362, r_{2,2} = 34,$$

y

$$y_0 = 337, x_0 = 0, y_1 = 337, x_1 = 1,$$

obteniendo  $m_2 = 0 + 1 \cdot 3 = 3$ .

Finalmente, en la tercera vuelta  $p_3 = 11$  y  $e_3 = 1$ . La raíces son

$$r_{3,0} = 1, r_{3,1} = 290, r_{3,2} = 333, r_{3,3} = 99,$$

$$r_{3,4} = 126, r_{3,5} = 16, r_{3,6} = 273, r_{3,7} = 167,$$

$$r_{3,8} = 393, r_{3,9} = 31, r_{3,10} = 256$$

y el coeficiente

$$y_0 = 337, x_0 = 9,$$

de donde  $m_3 = 9$ .

Finalmente, el logaritmo buscado es la solución al sistema de ecuaciones

$$m \equiv 3 \pmod{4}$$

$$m \equiv 3 \pmod{9}$$

$$m \equiv 9 \pmod{11},$$

es decir,  $m = 75$ .

### 4.1.3 Cálculo de índices en cuerpos primos

Sea  $\mathbb{F}_p = \mathbb{Z}_p$  con  $p$  primo. Sea  $b$  un elemento primitivo módulo  $p$ . Sea  $2 \leq B < p$ . La fase de precálculo consiste en calcular  $\log_b r$  para todo primo  $r \leq B$ . Para ello, se elige aleatoriamente  $1 \leq t \leq p-2$  hasta que  $b^t \bmod p$  es  $B$ -suave, es decir,

$$b^t \equiv \prod_{\substack{r \leq B \\ r \text{ primo}}} r^{m_{t,r}} \pmod{p}.$$

En consecuencia

$$t \equiv \sum_{\substack{r \leq B \\ r \text{ primo}}} m_{t,r} \log_b(r) \pmod{p-1}.$$

Repitiendo suficientes elecciones de  $t$ , obtendremos un sistema de ecuaciones lineales sobre  $\mathbb{Z}_{p-1}$  cuya solución son los valores buscados

$$\{\log_b(r) \mid r \leq B, r \text{ primo}\}.$$

Para calcular  $\log_b(y)$  con  $y \in \mathbb{Z}_p^*$ , volvemos a elegir valores aleatorios  $1 \leq t \leq p-2$  hasta encontrar uno que satisfaga

$$yb^t \equiv \prod_{\substack{r \leq B \\ r \text{ primo}}} r^{m_r} \pmod{p}.$$

Una vez encontrado,

$$\log_b(y) \equiv -t + \sum_{\substack{r \leq B \\ r \text{ primo}}} m_r \log_b(r) \pmod{p-1}.$$

La usabilidad de este algoritmo requiere que  $p$  no sea demasiado grande para que haya suficientes elementos  $B$ -suaves en  $\mathbb{Z}_p$ .

**Ejemplo 4.6.** Repetimos el cálculo del Ejemplo 4.5 mediante el cálculo de índices. Recordemos que  $p = 397$  y  $b = 5$ . Tomamos  $B = 5$ . Tenemos, por tanto, que calcular  $\log_5(2), \log_5(3), \log_5(5)$ . Elegimos valores aleatorios  $1 \leq t \leq 395$  tales que

$$5^t \equiv 2^{m_{t,2}} 3^{m_{t,3}} 5^{m_{t,5}} \pmod{397}.$$

Para  $t = 212$ ,  $5^t \equiv 90 = 2 \cdot 3^2 \cdot 5 \pmod{397}$ , lo que nos da una primera ecuación

$$212 \equiv 1 \log_5(2) + 2 \log_5(3) + 1 \log_5(5) \pmod{396}.$$

Para  $t = 360$ ,  $5^t \equiv 256 = 2^8 \pmod{397}$ , lo que nos da la ecuación

$$360 \equiv 8 \log_5(2) + 0 \log_5(3) + 0 \log_5(5) \pmod{396}.$$

Para  $t = 366$ ,  $5^t \equiv 225 = 3^2 \cdot 5^2 \pmod{397}$ , lo que nos da la ecuación

$$366 \equiv 0 \log_5(2) + 2 \log_5(3) + 2 \log_5(5) \pmod{396}.$$

Aunque tenemos tres ecuaciones, el determinante de los coeficientes,  $-16$ , no es una unidad módulo  $p-1 = 396$ , por lo que debemos continuar. Para  $t = 391$ ,  $5^t \equiv 288 = 2^5 \cdot 3^2 \pmod{397}$ , lo que nos da la cuarta ecuación

$$391 \equiv 5 \log_5(2) + 2 \log_5(3) + 0 \log_5(5) \pmod{396}.$$

Para  $t = 150$ ,  $5^t \equiv 27 = 3^3 \pmod{397}$ , lo que nos da la ecuación

$$150 \equiv 0 \log_5(2) + 3 \log_5(3) + 0 \log_5(5) \pmod{396}.$$

Para  $t = 246$ ,  $5^t \equiv 250 = 2 \cdot 5^3 \pmod{397}$ , lo que nos da la ecuación

$$246 \equiv 1 \log_5(2) + 0 \log_5(3) + 3 \log_5(5) \pmod{396}.$$

Para  $t = 151$ ,  $5^t \equiv 135 = 3^3 \cdot 5 \pmod{397}$ , lo que nos da la ecuación

$$151 \equiv 0 \log_5(2) + 3 \log_5(3) + 1 \log_5(5) \pmod{396}.$$

Llegados a este punto, comprobamos que

$$391 \equiv 5 \log_5(2) + 2 \log_5(3) + 0 \log_5(5) \pmod{396}$$

$$246 \equiv 1 \log_5(2) + 0 \log_5(3) + 3 \log_5(5) \pmod{396}$$

$$151 \equiv 0 \log_5(2) + 3 \log_5(3) + 1 \log_5(5) \pmod{396}$$

tiene solución única

$$\log_5(2) = 243, \log_5(3) = 182, \log_5(5) = 1.$$

Para calcular  $\log_5(337)$ , buscamos  $1 \leq t \leq 395$  tal que

$$337 \cdot 5^t \equiv 2^{m_2} 3^{m_3} 5^{m_5} \pmod{397}.$$

Por ejemplo,  $t = 260$  satisface

$$337 \cdot 5^{260} \equiv 200 = 2^3 \cdot 5^2 \pmod{397},$$

de donde

$$\begin{aligned} \log_5(337) &= -260 + 3 \log_5(2) + 0 \log_5(3) + 2 \log_5(5) \\ &= 471 \equiv 75 \pmod{396}. \end{aligned}$$

---

#### 4.1.4 Cálculo de índices en cuerpos finitos

Sea  $\mathbb{F}_q$   $q = p^n$ , y sea  $b \in \mathbb{F}_q^*$  un elemento primitivo. Supongamos que  $\mathbb{F}_q = \mathbb{F}_p[\alpha]/(f(\alpha))$  con  $f$  un polinomio irreducible de grado  $n$ . Identificamos los elementos de  $\mathbb{F}_q$  con polinomios en  $\alpha$  sobre  $\mathbb{F}_p$  de grado menor que  $n$ . Sea  $b' = b^{(q-1)/(p-1)}$ . Tenemos que

$$(b')^p = (b^{(q-1)/(p-1)})^p = b^{\frac{p(q-1)}{(p-1)}} = b^{q-1} b^{(q-1)/(p-1)} = b',$$

por lo que  $b' \in \mathbb{F}_p$  por ser un elemento invariante ante el automorfismo de Frobenius. De forma análoga,

$$(b')^{p-1} = b^{q-1} = 1$$

y si  $(b')^r = 1$  para un divisor propio  $r \mid p-1$ , obtenemos que

$$b^{r(q-1)/(p-1)} = 1$$

con  $\frac{r(q-1)}{p-1} \mid q-1$  un divisor propio, lo que contradice el que  $b$  es primitivo. Hemos demostrado que

$$b' = b^{(q-1)/(p-1)} \text{ es un elemento primitivo de } \mathbb{F}_p.$$

Como consecuencia, si podemos calcular logaritmos discretos en  $\mathbb{F}_p$  con respecto de  $b'$ , podemos calcular logaritmos discretos de elementos en  $\mathbb{F}_p \subseteq \mathbb{F}_q$  con respecto de  $b$ .

Sea  $B \subseteq \mathbb{F}_q^*$  el conjunto formado por todos los polinomios mónicos (irreducibles) en  $\mathbb{F}_p[\alpha]$  de grado menor que  $m < n$ . Se elige  $m$  de forma que  $|B|$  tiene un tamaño intermedio entre  $p$  y  $q$ .

Elegimos aleatoriamente  $1 \leq t \leq q-2$  y calculamos  $b^t$ , o equivalentemente,

$$c(\alpha) = b(\alpha)^t \bmod f(\alpha).$$



Comprobamos si

$$c(\alpha) = c_0 \prod_{\alpha(\alpha) \in B} a(\alpha)^{m_{c, \alpha}}$$

Esto puede hacerse dividiendo los polinomios de  $B$  entre  $c(\alpha)$  o factorizando  $c(\alpha)$  y observando si los factores están en  $B$ . Si  $c(\alpha)$  no es de esta forma repetimos el proceso para un nuevo  $t$ . Llegado el caso,

$$\log_b(c) - \log_b(c_0) \equiv \sum_{\alpha \in B} m_{c, \alpha} \log_b(a) \pmod{q-1}.$$

Puesto que  $\log_b(c) = t$ ,  $\log_b(c_0)$  se puede calcular y los valores  $m_{c, \alpha}$  son conocidos, tenemos una ecuación lineal módulo  $q-1$  cuyas incógnitas son  $\log_b(a)$ ,  $a \in B$ . Repetimos el proceso tantas veces como necesitemos para obtener suficientes ecuaciones independientes en  $\mathbb{Z}_{q-1}$  que nos permitan calcular  $\log_b(a)$ ,  $a \in B$ . Con esto concluye la fase de precálculo.

Observemos que si  $m$  es demasiado pequeño será difícil encontrar valores  $t$  tales que  $c = b^t$  descomponga como producto de elementos en  $B$ . Si  $m$  es demasiado grande, el tamaño de  $B$  será prohibitivamente grande para manejar los sistemas de ecuaciones. Sirva como ejemplo, que para  $p = 2$  y  $n = 127$ , suele tomarse un valor  $m = 17$ , lo que nos lleva a  $|B| = 16510$ .

Para calcular  $\log_b(y)$ , se eligen aleatoriamente valores  $1 \leq t \leq q-2$ , hasta que

$$y_1(\alpha) = y_0 \prod_{\alpha \in B} a(\alpha)^{m_\alpha}$$

donde  $y_1 = yb^t$ . Llegados a este punto,

$$\begin{aligned}\log_b(y) &= \log_b(y_1) - t \\ &= \log_b(y_0) + \sum_{a \in B} m_a \log_b(a) - t \pmod{q-1},\end{aligned}$$

por lo que calculamos el logaritmo.

*Ejemplo 4.7.* Sea  $p = 2$  y  $n = 6$ , con lo que  $q = 2^6 = 64$ . Representamos  $\mathbb{F}_{64} = \mathbb{F}_2[\alpha] / (\alpha^6 + \alpha^4 + \alpha^3 + \alpha + 1)$ . Sea  $b = \alpha^4 + \alpha^3 + \alpha^2$ . Como  $b^{21} = \alpha^3 + \alpha^2 + \alpha$  y  $b^9 = \alpha^5 + \alpha^4 + \alpha^2 + 1$ , tenemos que  $b$  es un elemento primitivo. Sea  $y = \alpha^5 + \alpha^2$ . Vamos a calcular  $\log_b y$ . Lo primero que tenemos que observar es que  $b' = b^{63} = 1$ , luego  $\log_b(1) = 63 \log_b(b') = 0$ . Tomamos  $m = 2$ , por lo que  $B = \{x, x+1, x^2+x+1\}$ . Vamos a calcular  $\log_b(\alpha)$ ,  $\log_b(\alpha+1)$ ,  $\log_b(\alpha^2+\alpha+1)$ .

Para  $t = 49$ , tenemos que

$$b^{49} = \alpha(\alpha+1)^2(\alpha^2+\alpha+1),$$

de dónde

$$49 = \log_b(\alpha) + 2 \log_b(\alpha+1) + \log_b(\alpha^2+\alpha+1) \pmod{63}.$$

Para  $t = 7$ , tenemos que

$$b^7 = (\alpha+1)^5,$$

de dónde

$$7 = 0 \log_b(\alpha) + 5 \log_b(\alpha+1) + 0 \log_b(\alpha^2+\alpha+1) \pmod{63}.$$

Para  $t = 42$ , tenemos que

$$b^{42} = (\alpha + 1)^3,$$

de dónde

$$42 = 0 \log_b(\alpha) + 3 \log_b(\alpha + 1) + 0 \log_b(\alpha^2 + \alpha + 1) \pmod{63}.$$

Estas tres ecuaciones describen un sistema de ecuaciones lineales sobre  $\mathbb{Z}_{63}$  descrito como

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 5 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} \log_b(\alpha) \\ \log_b(\alpha + 1) \\ \log_b(\alpha^2 + \alpha + 1) \end{pmatrix} = \begin{pmatrix} 49 \\ 7 \\ 42 \end{pmatrix}.$$

La matriz de coeficientes no tiene inversa módulo 63, por lo que hay que buscar otra ecuación. Para  $t = 44$ , tenemos que

$$b^{44} = \alpha^3(\alpha^2 + \alpha + 1),$$

de dónde

$$44 = 3 \log_b(\alpha) + 0 \log_b(\alpha + 1) + \log_b(\alpha^2 + \alpha + 1) \pmod{63},$$

El sistema se ha ampliado a

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 5 & 0 \\ 0 & 3 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} \log_b(\alpha) \\ \log_b(\alpha + 1) \\ \log_b(\alpha^2 + \alpha + 1) \end{pmatrix} = \begin{pmatrix} 49 \\ 7 \\ 42 \\ 44 \end{pmatrix},$$

que sí tiene solución única. Para comprobar esto podemos quitar la tercera ecuación, ya que 5 tiene inverso módulo 63 pero 3 no lo tiene, por lo que la matriz en  $\mathbb{Z}_{63}$

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 5 & 0 \\ 3 & 0 & 1 \end{pmatrix}$$

tiene inversa, y el sistema

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 5 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} \log_b(\alpha) \\ \log_b(\alpha + 1) \\ \log_b(\alpha^2 + \alpha + 1) \end{pmatrix} = \begin{pmatrix} 49 \\ 7 \\ 44 \end{pmatrix}$$

tiene solución única, siendo dicha solución

$$\begin{pmatrix} \log_b(\alpha) \\ \log_b(\alpha + 1) \\ \log_b(\alpha^2 + \alpha + 1) \end{pmatrix} = \begin{pmatrix} 43 \\ 14 \\ 41 \end{pmatrix}.$$

Para continuar con el algoritmo, para  $t = 2$  tenemos que

$$y_1 = yb^2 = \alpha^3(\alpha + 1),$$

lo que se traduce en

$$\begin{aligned} \log_b y + 2 &= 3 \log_b(\alpha) + \log_b(\alpha + 1) + 0 \log_b(\alpha^2 + \alpha + 1) \pmod{63} \\ &= 3 \times 43 + 13 \pmod{63} = 17, \end{aligned}$$

de donde

$$\log_b y = 15.$$

### Protocolo de Diffie-Hellman

En esta sección  $G = \mathcal{U}(\mathbb{Z}_p) = \mathbb{Z}_p \setminus \{0\}$  con  $p$  primo, y  $g$  es un generador de  $G$ .

**Conjetura** (Diffie y Hellman). *Dados  $g^a \bmod p$  y  $g^b \bmod p$ , calcular  $g^{ab} \bmod p$  es computacionalmente equivalente a calcular uno de los logaritmos  $a = \log_g g^a \bmod p$  o  $b = \log_g g^b \bmod p$ .*

Basado en esta conjetura, el protocolo de Diffie y Hellman es un protocolo de intercambio de claves para ser empleado en un criptosistema simétrico.

**Parámetros.** Se elige un primo  $p$  tal que el problema del logaritmo discreto sea difícil en  $\mathcal{U}(\mathbb{Z}_p)$ . Como mínimo necesitamos que  $p$  sea grande para que el Algoritmo 3 no sea efectivo, y que  $p - 1$  tenga factores grandes para que sea ineficaz el Algoritmo 4.

El siguiente paso es la elección de un elemento primitivo de  $\mathbb{Z}_p$ , es decir, un generador del grupo multiplicativo  $\mathcal{U}(\mathbb{Z}_p) = \mathbb{Z}_p \setminus \{0\}$ .

**Lema 4.8.**  $g \in \mathbb{Z}_p \setminus \{0\}$  es primitivo si y solo si

$$g^{(p-1)/p_i} \not\equiv 1 \pmod{p}$$

para cada factor primo  $p_i$  de  $p - 1$ .

**Demostración.** Por el Teorema de Lagrange, si  $r$  es el menor exponente positivo tal que  $g^r \equiv 1 \pmod{p}$ , entonces  $r \mid p - 1$ . Por el Teorema Fundamental de la Aritmética, si  $r \neq p - 1$  entonces  $r \mid \frac{p-1}{p_i}$  para algún divisor primo  $p_i \mid p - 1$ , lo que implica que  $g^{(p-1)/p_i} \equiv 1 \pmod{p}$ .  $\square$

Este lema nos proporciona un método eficiente para comprobar si un elemento es primitivo. Para encontrar un elemento primitivo vamos a seleccionar aleatoriamente elementos de  $\mathbb{Z}_p \setminus \{0\}$  hasta encontrar uno.

**Lema 4.9.** Hay  $\varphi(p-1)$  elementos primitivos en  $\mathbb{Z}_p \setminus \{0\}$ . Sea  $\rho = \varphi(p-1)/(p-1)$ . La probabilidad de encontrar un elemento primitivo después de  $\ell$  intentos es  $1 - (1 - \rho)^\ell$ .

*Demostración.* Como  $\mathcal{U}(\mathbb{Z}_p)$  es cíclico, hay al menos un elemento primitivo  $g$ , que tiene orden  $p-1$ . Cualquier potencia de  $g$  cuyo exponente sea primo con  $p-1$  es de nuevo un elemento primitivo, y hay  $\varphi(p-1)$  tales exponentes. Por otra parte, una potencia con exponente no primo con  $p-1$  no puede ser elemento primitivo.

La segunda parte del Lema es inmediata dado que los procesos de elección son independientes.  $\square$

En [4, Theorem 328] se demuestra que  $\varphi(p-1)/(p-1)$  está asintóticamente acotado por abajo por un múltiplo constante de  $\log \log(p-1)$ . Si  $p-1$  tiene un divisor primo  $r > \sqrt{p}$ , tenemos que

$$\begin{aligned}\varphi(p-1) &= \varphi\left(\frac{p-1}{r}\right) \varphi(r) \\ &= \varphi\left(\frac{p-1}{r}\right) (r-1) \\ &\geq r-1,\end{aligned}$$

luego

$$\frac{\varphi(p-1)}{p-1} \geq \frac{r-1}{p-1} \geq \frac{\sqrt{p}-1}{p-1} \approx \frac{1}{\sqrt{p}},$$

lo que permite encontrar con cierta rapidez un elemento primitivo. Concretamente, si  $\ell \geq \frac{1}{2} \log_2(p)$ , la probabilidad de encontrar un elemento primitivo después de  $\ell$  intentos es mayor que  $\frac{1}{2}$ .

**Generación de claves pública/privada.** Una vez establecida la pareja de parámetros  $p, g$ , un usuario elige aleatoriamente un valor  $2 \leq x \leq p - 2$ , y hace público el valor  $g^x \bmod p$ , manteniendo  $x$  en secreto.

**Clave compartida** Supongamos que Alicia tiene por pareja de claves  $(a, g^a \bmod p)$  y Bob  $(b, g^b \bmod p)$ . Ambos calculan

$$g^{ab} \bmod p = (g^a)^b \bmod p = (g^b)^a \bmod p,$$

que será su clave compartida. Si la Conjetura de Diffie y Hellman es cierta, un atacante debe calcular un logaritmo discreto en  $\mathcal{U}(\mathbb{Z}_p)$  si quiere conocer  $g^{ab} \bmod p$  a partir de  $g^a \bmod p$  y  $g^b \bmod p$ .

---

4.3

### Criptosistema de ElGamal

Este criptosistema emplea el logaritmo discreto como función unidireccional. Sea  $\mathbb{F}_q$  un cuerpo con  $q$  elementos. Recordemos que  $\mathcal{U}(\mathbb{F}_q) = \mathbb{F}_q \setminus \{0\}$  es un grupo cíclico de orden  $q - 1$ .

**Generación de claves.** El usuario Alicia toma aleatoriamente  $1 < a < q - 1$ , calcula  $g^a \in \mathbb{F}_q$  y hace público  $(\mathbb{F}_q, g, g^a)$ , manteniendo  $a$  como clave privada.

**Algoritmo de cifrado.** El mensaje es  $m \in \mathbb{F}_q \setminus \{0\}$ , y el algoritmo

$$E_{g^a}(m) = (g^k, m(g^a)^k)$$

para un valor aleatorio  $1 < k < q - 1$ .

**Algoritmo de descifrado.** El descifrado es el siguiente,

$$D_a(x, y) = yx^{-a}.$$

**Teorema 4.10.**  $D_a(E_{g^a}(m)) = m$ .

*Demostración.* Si  $1 < k < q - 1$ ,

$$D_a(E_{g^a}(m)) = D_a(g^k, m(g^a)^k) = m(g^a)^k(g^k)^{-a} = m.$$

como queríamos. □

Como conocemos el orden del grupo  $\mathcal{U}(\mathbb{F}_q)$ , la igualdad

$$x^{-a} = x^{q-1-a}$$

nos permite calcular  $x^{-a}$  como potencia sin necesidad de calcular inversos. Lo usual es que  $q = p$  primo, por lo que  $\mathbb{F}_q = \mathbb{Z}_p$ . Las claves y algoritmos pueden reescribirse como

$$\begin{aligned} & (p, g, a, g^a \bmod p), \\ E_{g^a}(m) &= (g^k \bmod p, m(g^a)^k \bmod p), \\ D_a(x, y) &= yx^{p-1-a} \bmod p. \end{aligned}$$



Este cifrado es aleatorio, es decir, si ciframos dos veces el mismo mensaje con la misma clave obtenemos criptogramas distintos. Para descifrar no es necesario conocer el valor de  $k$ . El criptograma contiene el mensaje enmascarado  $mg^{ak}$  junto con un desenmascarador  $g^k$ .

Para averiguar  $m$  a partir del criptograma es necesario calcular  $g^{ak}$ , y un atacante conoce  $g^a$ , la clave pública, y  $g^k$ , la primera parte del criptograma. Según la conjetura de Diffie y Hellman, el atacante debe, por tanto, resolver un logaritmo discreto.

## 4.4

### Digital Signature Algorithm

Hay otra construcción criptográfica basada en el concepto de clave pública, el concepto de firma digital. Dada una pareja de claves ( $K, k$ ) pública–privada, un esquema de firma digital consta de un algoritmo de firma dependiente de la clave privada

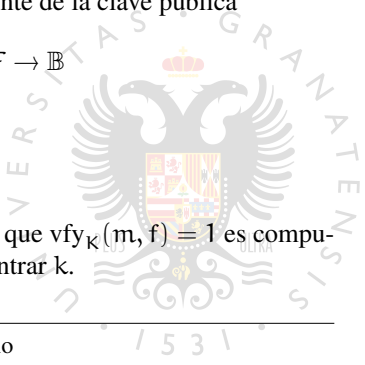
$$\text{sgn}_k : \mathcal{M} \rightarrow \mathcal{F}$$

y un algoritmo de verificación dependiente de la clave pública

$$\text{vfy}_K : \mathcal{M} \times \mathcal{F} \rightarrow \mathbb{B}$$

tales que

- $\text{vfy}_K(m, \text{sgn}_k(m)) = 1$
- Encontrar  $(m, f) \in \mathcal{M} \times \mathcal{F}$  tales que  $\text{vfy}_K(m, f) = 1$  es computacionalmente equivalente a encontrar  $k$ .



*Ejemplo 4.11.* Sean  $(n, e)$  y  $(p, q, d)$  las claves pública y privada de un criptosistema RSA. El siguiente es un esquema de firma digital:

$$\text{sgn}_{(p,q,d)}(m) = m^d \bmod n$$

$$\text{vfy}_{(n,e)}(m, f) = \begin{cases} 1 & \text{si } m \equiv f^e \bmod n, \\ 0 & \text{en otro caso.} \end{cases}$$

El DSA (Digital Signature Algorithm) fue propuesto por el NIST en 1991, y adoptado como DSS (Digital Signature Standard) en 1993. Desde entonces se han propuesto cuatro revisiones del mismo, la última en 2013 [7].

### Generación de claves

- Se elige un primo  $q$  de  $N$  bits.
- Se elige un primo  $p$  de  $L$  bits tal que  $p = 2kq + 1$ . Las combinaciones  $N, L$  permitidas son

L	1024	2048	2048	3072
N	160	224	256	256

- Elegimos  $h \in \mathbb{Z}_p^*$  tal que  $h^{(p-1)/q} \not\equiv 1 \bmod p$ . Llamamos  $g = h^{(p-1)/q} \bmod p$ .
- Elegimos  $1 \leq x \leq q - 1$  aleatoriamente. Calculamos  $y = g^x \bmod p$ .
- La clave privada (usada para firmar) es  $(p, q, g, x)$ .
- La clave pública (usada para verificar) es  $(p, q, g, y)$ .

### Proceso de firma

- El mensaje es  $m \in \mathbb{Z}_q$ . Se obtiene como el resultado de aplicar una función hash a un mensaje mayor.
- Seleccionamos aleatoriamente  $1 \leq k \leq q$ .
- Calculamos  $r = (g^k \bmod p) \bmod q$  y  $s = k^{-1}(m + rx) \bmod q$ . Si  $s = 0$  volvemos a seleccionar  $k$ , pero es muy improbable que ocurra.
- El mensaje firmado es  $(m, r, s)$ .

### Verificación

- El receptor conoce el mensaje firmado  $(m, r, s)$  y la clave pública  $(p, q, g, y)$ .
- Calculamos  $t = s^{-1} \bmod q$  y  $v = ((g^m y^r)^t \bmod p) \bmod q$ .
- La verificación es afirmativa si y sólo si  $v = r$ .

**Proposición 4.12.** *Si la firma es correcta,  $v = r$ .*

*Demostración.*

$$\begin{aligned}
 v &= ((g^m y^r)^t \bmod p) \bmod q \\
 &= (g^{mt} g^{rxt} \bmod p) \bmod q \\
 &= (g^{(m+rx)t} \bmod p) \bmod q \\
 &= (g^k \bmod p) \bmod q \\
 &= r
 \end{aligned}$$

donde los exponentes se calculan módulo  $q$ . □

## Seguridad de DSA

- Un cálculo eficiente de logaritmos discretos permite calcular la clave privada a partir de la clave pública, y por tanto generar todas las firmas deseadas.
- Dados  $m, r$ , encontrar  $s$  tal que la terna  $(m, r, s)$  es una firma correcta es equivalente a resolver la ecuación

$$r^s \equiv (g^m y^r) \pmod{p},$$

que es un logaritmo discreto.

- Dado  $m$ , para encontrar una firma correcta  $m, r, s$  podemos, de nuevo, tratar de resolver la ecuación

$$r^s \equiv (g^m y^r) \pmod{p},$$

que no es un logaritmo discreto, pero no es fácil de resolver de forma eficiente.

- Es fácil, para un atacante fabricar una terna  $m, r, s$  que sea una firma correcta, tal y como se describe en el Ejercicio 4.3. Por ello, para que el algoritmo DSA sea considerado un algoritmo de firma digital seguro es necesario utilizarlo con las llamadas funciones Hash.

---

## Ejercicios de logaritmo discreto

**Ejercicio 4.1.** Intenta calcular los siguientes logaritmos discretos mediante alguno de los métodos descritos.

$$\log_{118}(2) \bmod 127,$$

$$\log_{28}(115) \bmod 379,$$

$$\log_{186}(87) \bmod 223,$$

$$\log_{97}(103) \bmod 127,$$

$$\log_{240}(20) \bmod 347$$

**Ejercicio 4.2.** Sea  $\mathbb{F}_{81} = \mathbb{Z}_3[\alpha]_{\alpha^4 - \alpha^3 - 1}$ , donde representamos  $\mathbb{Z}_3 = \{-1, 0, 1\}$ . Comprueba que  $\alpha$  es un generador de  $\mathbb{F}_{81}^*$ . Calcula  $\log_{\alpha}(\alpha + 1)$  usando el cálculo de índices. Deduce del valor anterior que  $\alpha + 1$  no es un generador de  $\mathbb{F}_{81}$ .

**Ejercicio 4.3.** Sea  $(p, q, g, y)$  la clave pública de un esquema de firma DSA. Sean  $0 < a, b < q$ ,  $r = (g^a y^b \bmod p) \bmod q$ ,  $s = rb^{-1} \bmod q$  y  $m = ar b^{-1} \bmod q$ . Comprueba que la firma  $(m, r, s)$  es una firma correcta.

**Ejercicio 4.4.** ¿Por qué no es bueno emplear el grupo aditivo  $\mathbb{Z}_n$  para diseñar un protocolo tipo Diffie-Hellmann?

**Ejercicio 4.5.**

1. Genera parámetros  $p, g$  donde  $p$  es un primo tal que  $2^9 \leq p < 2^{10}$ , y  $g$  es un generador de  $\mathbb{F}_p^*$ .

2. Genera dos parejas de claves pública/privada usando los parámetros anteriores para dos usuarios y simula un intercambio de claves de Diffie y Hellman, calculando la clave compartida desde el punto de vista de cada uno de los usuarios.
3. Cifra el mensaje  $m = 0b101101100$  mediante el criptosistema del ElGamal usando una de las dos claves públicas anteriores.
4. Descifra el criptograma obtenido.



---

## Ejercicios de evaluación de logaritmo discreto

**Ejercicio.** Los parámetros de un criptosistema de ElGamal son  $p = 211$  y  $g = 3$ , es decir, el criptosistema está diseñado en el cuerpo  $\mathbb{F}_{211} = \mathbb{Z}_{211}$  y tomamos como generador de  $\mathbb{F}_{211}^*$ ,  $g = 3$ . La clave pública empleada es  $3^a = 109 \pmod{211}$ . Descifra el criptograma  $(154, \text{dni} \pmod{211})$ , donde dni es el número de tu DNI. Para calcular los logaritmos discretos necesarios emplea dos de los métodos descritos en la teoría.



## Bibliografía

- [1] Gregory V. Bard. *Algebraic Cryptanalysis*. Springer Science and Business Media, 2009.
- [2] Hans Delfs and Helmut Knebl. *Introduction to Cryptography*. Information Security and Cryptography. Springer-Verlag Berlin Heidelberg, 2015.
- [3] Andreas Enge. *Elliptic curves and their applications to cryptography. An Introduction*. Kluwer Academic Publishers, 1999.
- [4] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, fourth edition, 1960.
- [5] Nathan Jacobson. *Basic Algebra: I*. W.H. Freeman & Company, second edition, 1985.
- [6] Neal Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 2 edition, 1994.
- [7] National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS)*, July 2013.



- [8] Harald Niederreiter and Arne Winterhof. *Applied Number Theory*. Springer International Publishing, 2015.
- [9] Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer International Publishing, 2016.
- [10] Joachim von zur Gathen. *CryptoSchool*. Springer-Verlag Berlin Heidelberg, 2015.

