Introducción a la Criptología

Ángel del Río Mateos

January 27, 2021

Contenidos

Capítulo 1

Criptosistemas simétricos o de clave privada

1.1 Criptografía

La *criptografía* es la ciencia de representar información de forma opaca para que sólo los agentes autorizados (personas o dispositivos diversos) sean capaces de desvelar el mensaje oculto. El proceso de ocultar la información se llama *cifrado*, pero a menudo también se le llama *encriptado* por influencia del inglés. El proceso de desvelarla se llama *descifrado* o *desencriptado*. El concepto de criptosistema modela los procesos de cifrado y descifrado.

Un *criptosistema simétrico*, también llamado *de clave privada*, está formado por un conjunto K, cuyos elementos llamamos *claves* o *llaves*, y una regla que asocia dos aplicaciones a cada clave $k \in K$:

$$c_k : M_k \to C_k \qquad d_k : C_k \to M_k,$$

de forma que

$$d_k(c_k(x)) = x, \text{ para todo } x \in M_k.$$

En la práctica el protocolo criptográfico también incluye un algoritmo generador de claves, es decir uno que tiene como salida un elemento de K, pero nosotros no vamos a tener en cuenta esta parte del criptosistema.

Extendemos las aplicaciones c_k a $M_k^\infty = \cup_{n\geq 1} M_k^n$ y d_k a $M_k^\infty = \cup_{n\geq 1} M_k^n$ poniendo

$$\begin{aligned} c_k(x_1 \dots x_n) &= c_k(x_1)c_k(x_2)\dots c_k(x_n); \quad x_1, \dots, x_n \in M_k \\ d_k(y_1 \dots y_n) &= d_k(y_1)d_k(y_2)\dots d_k(y_n); \quad y_1, \dots, y_n \in C_k. \end{aligned}$$

Obsérvese que representamos los elementos de M^n como concatenación de elementos de M.

Utilizaremos la siguiente terminología para una clave $k \in K$:

$$\begin{aligned} \text{Elementos de } M_k^\infty &= \textit{Mensajes en claro}; \\ \text{Elementos de } C_k^\infty &= \textit{Mensajes cifrados o encriptados}; \\ c_k &= \textit{Función de cifrado} \text{ o } \textit{función de encriptado}; \\ d_k &= \textit{Función de descrifrado} \text{ o } \textit{función de desencriptado} \end{aligned}$$

Un mensaje en claro o cifrado diremos que es *básico* si tiene longitud 1 . Sin embargo, en muchas situaciones diremos mensajes para referirnos a mensajes básicos, bien en claro o cifrados. Esperamos que esto no cause confusión porque quede claro en el contexto.

Frecuentemente M_k es el mismo conjunto para todas las claves y lo mismo ocurre con los C_k. En ese caso ponemos $M = M_k$ y $C = C_k$.

Ejemplos 1.1 *Criptosistemas*
Sea A un conjunto finito y denotemos por S_A el conjunto de permutaciones de los elementos de un conjunto finito A.

(1) *Sustitución.* Tomamos como conjunto de claves $K = S_A$, como conjuntos de mensajes básicos $M = N = A$ y como funciones de cifrado y descifrado:

$$c(\sigma, x) = \sigma(x) \quad \text{y} \quad d(\sigma, y) = \sigma^{-1}(y).$$

(2) *Reordenamiento.* Ponemos $K = \cup_{n\geq 2} S_n$, donde $S_n = S_{\{1,2,\ldots,n\}}$. Si la clave σ está en S_n, entonces ponemos $M_\sigma = C_\sigma = A^n$ y ciframos reordenando las posiciones de los símbolos de los mensajes en claro. Más precisamente:

$$c_\sigma(x_1 \ldots, x_n) = x_{\sigma^{-1}(1)} \ldots x_{\sigma^{-1}(n)} \quad \text{y} \quad d_\sigma(y_1 \ldots, y_n) = y_{\sigma^{-1}(1)} \ldots y_{\sigma^{-1}(n)}.$$

En la práctica casi todos los protocolos criptográficos son combinaciones de los de los Ejemplos 1.1. Sin embargo, en estos criptosistemas no se dice nada sobre las permutaciones elegidas. La naturaleza de estas permutaciones es lo que hace un criptosistema bueno o malo. La bondad de un criptosistema depende de que satisfaga las siguientes condiciones:

- *Rapidez de los cálculos.* Es importante disponer de un algoritmo eficiente (polinomial, con exponente pequeño) para calcular $c_k(x)$ y $d_k(y)$.
- *Seguridad.* Debe ser difícil descubrir un valor concreto de x a partir del valor de $c_k(x)$ sin conocer k.

Las nociones de "algoritmo eficiente", "difícil de calcular" y "tiempo razonable" son ambiguas. En la Sección 1.7 daremos conceptos más precisos. De momento, veamos ejemplos más concretos.

Criptosistema de Cesar

Vamos a identificar las 27 letras mayúsculas del alfabeto español asignando a cada una de ellas un número del 0 al 26, de forma que estamos identificando los elementos de A con los de $\mathbb{Z}_{27}$ usando la siguiente tabla:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13
$\tilde{N}$	O	P	Q	R	S	T	U	V	W	X	Y	Z	
14	15	16	17	18	19	20	21	22	23	24	25	26	

(1.1)

Ponemos $K = M = C = \mathbb{Z}_{27}$ y

$$\begin{aligned} c_k(x) &= (x + k \mod 27) \\ d_k(x) &= (x - k \mod 27). \end{aligned}$$

Por ejemplo, supongamos que la clave es la letra W, que corresponde al número 23 y que queremos cifrar el mensaje "ésta es la primera vez que vamos a cifrar un mensaje". En primer lugar obsérvese que nuestro alfabeto sólo contiene letras mayúsculas sin acentuar, con lo que el mensaje a cifrar es:

ESTAESLAPRIMERAVEZQUEVAMOSACIFRARUNMENSAJE

Ahora tenemos que convertir el mensaje en una lista de números de acuerdo a la tabla anterior:

4,19,20,0,4,19,11,0,16,18, 8,12,4,18,0,22,4,26,17,21,
4,22,0,12,15,19,0,2,8,5,18,0,18,21,13,12,4,13,19,0,9,4

Después tenemos que sumar a cada uno de estos números la clave, es decir sumamos 23 módulo 27, o lo que es lo mismo restamos 4 módulo 27.

0,15,16,23,0,15,7,23,12,14, 4,8,0,14,23,18,0,22,13,17,
0,18,23,8,11,15,23,25,4,1,14,23,14,17,9,8,0,9,15,23,5,0

Finalmente tenemos que sustituir los números por las letras correspondientes y obtendremos el mensaje cifrado.

AOPWAOHWMÑEIAÑWRAVNQARWILOWYEBÑWÑQJIAJOWFA

Criptosistema de Vigenère

Es similar al de Cesar con el alfabeto español identificado con $\mathbb{Z}_{27}$, pero $K = \mathbb{Z}_{27}^{\infty}$ y si $k \in \mathbb{Z}_{27}^{d}$ entonces $M_k = C_k = \mathbb{Z}_{27}^{d}$ y

$$\begin{aligned} c_{k_1 k_2 \ldots k_d}(x_1 x_2 \ldots x_d) &= (x_1 + k_1)(x_2 + k_2) \ldots (x_d + k_d) \\ d_{k_1 k_2 \ldots k_d}(x_1 x_2 \ldots x_d) &= (x_1 - k_1)(x_2 - k_2) \ldots (x_d - k_d). \end{aligned}$$

donde todas las operaciones se hacen en $\mathbb{Z}_{27}$. Es decir se divide el mensaje en bloques de longitud d y se trabaja como en el Criptosistema de Cesar usando la clave k_i para los símbolos en posiciones j con $j \equiv i \mod d$.

En realidad el criptosistema de César y Vigenère se pueden considerar como casos particulares de otros más generales. Simplemente hemos restringido la presentación a 27 símbolos para hacer más simple la explicación. Sin embargo el conjunto de símbolos puede ser cualquiera (letras mayúsculas y minúsculas, comas, espacios, etc), y podemos siempre identificar el conjunto de símbolos con $\mathbb{Z}_n$, donde n es el cardinal del conjunto de símbolos. De esta forma podemos obtener versiones del Criptosistema de César y del de Vigenère con un conjunto mayor de símbolos. Por otro lado, se puede cambiar $\mathbb{Z}_n$ por cualquier otro grupo sin que cambie de forma esencial la naturaleza de estos dos criptosistemas. Nos referiremos a este grupo como plataforma. Por ejemplo, en lugar de identificar los símbolos del alfabeto con los elementos

de $\mathbb{Z}_n$ los podemos identificar con los elementos de un grupo finito y utilizar la operación del grupo para hacer las operaciones. No es esencial que la aplicación que asocia a un símbolo un elemento del grupo plataforma sea suprayectiva, aunque si que es necesario que sea inyectiva.

Por otro lado, los símbolos pueden no ser elementos de un alfabeto natural sino combinaciones de varios símbolos. Es decir, podemos elegir una cierta longitud, pongamos n, e identificar listas de n símbolos en nuestro alfabeto, con el grupo aditivo de $\mathbb{Z}_{b^n}$, donde b es el cardinal del alfabeto. Por ejemplo, supongamos que utilizamos el mismo alfabeto de antes, es decir, las letras mayúsculas del alfabeto en español y que las queremos agrupar en bloques de longitud cuatro. Entonces podemos identificar estas sucesiones con los elementos de $\mathbb{Z}_{27^4}$ mediante la siguiente regla:

$$AAAA = 0, AAAB = 1, \ldots, AAAAZ = 26, AABA = 27, AABB = 28, \ldots, ZZZZ = 27^4 - 1.$$

Obsérvese que el bloque $X_1X_2X_3X_4$ se identificará con $x_1 \cdot 27^3 + x_2 \cdot 27^2 + x_3 \cdot 27 + x_4$, donde x_i es el número identificado con la letra X_i. Por ejemplo, "MESA" se identificaría con

$$12 \cdot 27^3 + 4 \cdot 27^2 + 19 \cdot 27 + 0 = 239625.$$

De esta forma $M = C = \mathbb{Z}_{27^4}$ y podríamos utilizar cualquiera de los criptosistemas anteriores de forma similar.

En resumen, podemos establecer diferentes formas de asociar los mensajes básicos con los elementos de un grupo G. Todo esto no tiene ninguna naturaleza criptográfica, es simplemente la codificación de la información, de forma que podemos identificar el grupo plataforma G con los tres conjuntos fundamentales del protocolo criptográfico: $K = M = C = G$. Lo que tiene la naturaleza criptográfica son las funciones de cifrado y descifrado, que en este caso toma la forma

$$c_k(x) = kx, \quad \text{y} \quad d_k(y) = k^{-1}y.$$

Otra cosa importante, es que la naturaleza del grupo plataforma no es indiferente, pues para que podamos considerar el criptosistema como tal, necesitamos disponer de algoritmos eficientes para calcular las funciones de cifrado y descifrado y queremos que, sin el conocimiento de la clave sea difícil descifrar. Así por ejemplo $(\mathbb{Z}_n, +)$, $(\mathbb{Z}_n^*, \cdot)$ y los grupos aditivo y multiplicativo de un cuerpo finito o el grupo aditivo de un espacio vectorial sobre un cuerpo finito cumplen la primera condición pues la aritmética en estos grupos es sencilla.

Veamos un último ejemplo de criptosistema simétrico sencillo.

Criptosistemas afines

Sea n un entero positivo y pongamos $M = C = \mathbb{Z}_n$ y $K = \mathbb{Z}_n^* \times \mathbb{Z}_n$, donde $\mathbb{Z}_n^*$ representa el conjunto de elementos inversibles de $\mathbb{Z}_n$, es decir, los números naturales menores que n y coprimos con n. Las funciones de cifrado y descifrado son

$$\begin{array}{rcl} c_{(a,b)}(x) & = & (ax + b \mod n) \\ d_{(a,b)}(x) & = & (a^{-1}(x - b) \mod n) \end{array} \tag{1.2}$$

donde a^{-1} representa el inverso de a en $\mathbb{Z}_n$.

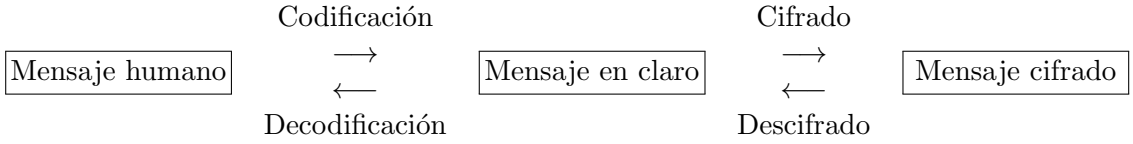
Más generalmente, si R es un anillo y R^* denota el grupo de las unidades de R, podemos poner $K = R^* \times R$, $M = C = R$ y utilizar las mismas funciones de cifrado y descifrado: $c_{(a,b}(x) = ax + b$ y $d_{(a,b)}(x) = a^{-1}(x - b)$, con operaciones en el anillo plataforma R.

Otra alternativa es tomar $K = \mathrm{GL}_d(R) \times R^d$, donde $\mathrm{GL}_d(R)$ denota el conjunto las matrices invertibles $d \times d$ con entradas en R. Entonces, usando multiplicación matricial elcriptosistema afín viene dado por

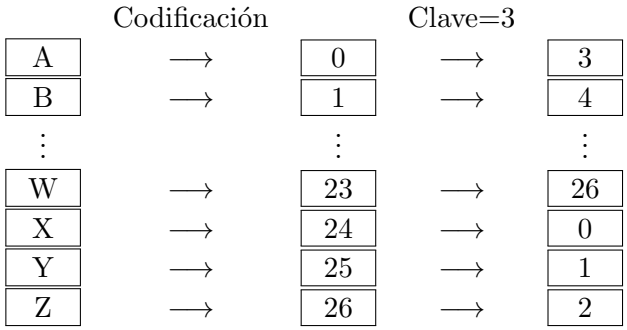
$$\begin{aligned} c_{(A,b)}(x) &= Ax + b \\ d_{(A,b)}(x) &= A^{-1}(x - b) \end{aligned}$$

Más generalmente, se puede fijar como conjunto de mensajes básicos, un R-módulo $M = C$ y elegir como conjunto de claves $K = \mathrm{Aut}(M) \times M$, donde $\mathrm{Aut}(M)$ es el grupo de los automorfismos de M, y como funciones de cifrado y descifrado $c((f, b), x) = f(x) + b$ y $d((f, b), y) = f^{-1}(y - b)$. En este caso la plataforma está formada por un anillo y un módulo y, de nuevo, la naturalezas algebraicas de las plataformas son importantes.

Cerramos esta sección con un esquema sencillo del proceso criptográfico:



En el proceso de codificación, transformamos un texto en símbolos que puedan ser procesados por algoritmos. Por ejemplo, en el Criptosistema de César los mensajes humanos son las letras de un alfabeto que las codificamos convirtiéndolas en números.



También podríamos hacer bloques de una longitud determinada, por ejemplo 4, y convertirlas en números considerando los símbolos del alfabeto original como los guarismos de un sistema de numeración, con lo que las palabras de una determinada longitud se interpretan como números de esa longitud en base b, donde b es el número de símbolos del alfabeto.

	Codificación		Clave=300	
AAAA	$\longrightarrow$	0	$\longrightarrow$	300
AAAB	$\longrightarrow$	1	$\longrightarrow$	301
$\vdots$		$\vdots$		$\vdots$
AAAZ	$\longrightarrow$	26	$\longrightarrow$	326
AABA	$\longrightarrow$	27	$\longrightarrow$	327
$\vdots$		$\vdots$		$\vdots$
BEYC	$\longrightarrow$	23276	$\longrightarrow$	23576
$\vdots$		$\vdots$		$\vdots$
ZZZZ	$\longrightarrow$	$27^4 - 1 = 531440$	$\longrightarrow$	299

Con este ejemplo el proceso de codificación de BEYC tiene dos pasos: Primero BEYC se convierte en la lista de números $(1, 4, 26, 2)$ y después convertimos este número en $2 + 26 \cdot 27 + 4 \cdot 27^2 + 27^3 = 23276$. Después cifamos sumando la clave. En el proceso de descifrado deberíamos aplicar el proceso inverso primero descifrando y después convirtiendo el mensaje en claro en un mensaje humano. Por ejemplo, supongamos que el mensaje cifrado es 25936. Desciframos restanto 300, con lo que obtendríamos 25636. Para decodificarlo tenemos que escribir este número en base 27 dividiendo sucesivamente por 27 y quedándonos con los restos:

$$\begin{aligned} 25636 &= 949 \cdot 27 + \mathbf{13} \\ 949 &= 35 \cdot 27 + \mathbf{4} \\ 35 &= \mathbf{1} \cdot 27 + \mathbf{8} \end{aligned}$$

Luego 25636 corresponde con la lista $(1, 8, 4, 13)$ que, de acuerdo con la tabla (1.1) corresponde con la palabra BIEN.

A nosotros nos importa poco el proceso de codificación y nos interesaremos más de los proceso de cifrado y descifrado. Sin embargo, es importante tener en cuenta que en el proceso de codificación se debe elegir una buena plataforma para hacer las cuentas del encriptado. En los ejemplos más sencillos la plataforma era $\mathbb{Z}_n$ para un número n; en los más complicados era otra estructura algebraica más abstracta. Pero lo importante es tener métodos de cálculo eficientes. Por ejemplo en $\mathbb{Z}_n$ las operaciones se reducen a operaciones aritméticas sencillas: sumas, multiplicaciones y divisiones. No hay ningún problema en trabajar con cuerpos finitos ya que veremos métodos de cálculo rápidos con dichos cuerpos. Tampoco habría grandes problemas en utilizar como plataforma anillos de matrices de cuerpos finitos o anillos de polinomios de dichos cuerpos, ya que las operaciones con cuerpos finitos son fácilmente transferibles a estos otros anillos. Se podrían también utilizar anillos de naturaleza discreta como el de los números enteros o el cuerpo de los números racionales, así como extensiones finitas de este. Sin embargo, no se pueden utilizar plataformas en las que no se puedan describir los elementos de una forma finita. Por ejemplo, el cuerpo de los números reales no es apropiado como plataforma.

1.2 Criptoanálisis

Criptoanálisis es la ciencia (¡o arte!) de desvelar un mensaje oculto sin necesidad de la clave

con la que fue encriptado. La combinación de criptografía y criptoanálisis se llama *criptología*.

La posibilidad de éxito del criptoanalista depende de la cantidad de información que posea. Por ejemplo, imaginemos las siguientes situaciones:

(1) El criptoanalista cuenta con un mensaje cifrado de longitud suficientemente grande.

(2) El criptoanalista posee uno o varios mensajes sin cifrar con su correspondiente mensaje cifrado.

(3) El criptoanalista puede generar tantas parejas (m, c) de mensajes en claro como desee junto con su correspondiente mensaje cifrado.

Se supone que el criptoanalista conoce las funciones c y d de cifrado y descifrado y, o bien desea encontrar la clave utilizada para poder descifrar los mensajes, o bien simplemente quiere descifrar un mensaje concreto. El problema de descifrar los mensajes enviados con una clave se llama el *Problema de Ruptura de un Criptosistema*.

Veamos como realizar un criptoanálisis del Criptosistema de Cesar. Supongamos que estamos en la situación de menor información posible, es decir tenemos un mensaje cifrado con el Criptosistema de Cesar (en la versión en la que los mensajes básicos son las 27 letras del alfabeto español):

ÑWEWUEPKBNÑUKVKWMQKNÑMEIYWYVLBÑWYAERÑBYKMYBNKBVÑ

Suponemos que sabemos que el mensaje en claro es una frase en español. Como las letras más frecuentes son las vocales, las letras más frecuentes en el mensaje cifrado deben tener una gran intersección con las correspondientes a las vocales. Las dos letras más frecuentes en el mensaje anterior son la K y la Ñ, cada una de las cuales aparece 6 veces. Probablemente una de éstas es el mensaje básico cifrado de la A. Probamos a descifrar con estas letras y descubrimos que el mensaje obtenido descifrando con la K es

ENUNLUGARDELAMANCHADECUYONOMBRENOQUIEROACORDARME

Claramente es difícil que la clave no sea K.

Por tanto, aunque el Criptosistema de Cesar satisface la primera condición para ser un buen criptosistema (rapidez de cifrado y descifrado) no cumple la condición de seguridad. La razón básica de por qué el Criptosistema de César no es seguro es que el número de claves es muy pequeño. Hemos hecho un análisis de frecuencia de las letras para evitar probar con las 27 claves pero con la ayuda de ordenadores esto sería inmediato con lo que se rompería muy fácilmente. Un ataque consistente en probar todas las claves es lo que se llama *ataque por fuerza bruta* mientras que un análisis de frecuencias de los símbolos utilizados se suele llamar *ataque por análisis de frecuencias*.

El Criptosistema de Vigenère es sólo un poco más seguro. Mejor dicho su seguridad dependerá de la longitud de la palabra clave. Aunque no se conozca dicha longitud si no es muy grande, un análisis de frecuencia de los caracteres que aparecen en el mensaje a intervalos regulares rompería el criptosistema de Vigenère. Esto lo veremos más adelante.

Analicemos ahora el criptosistema lineal. En el caso en el que el conjunto de mensajes se identifica con $\mathbb{Z}_n$ y las claves son parejas $(a, b) \in \mathbb{Z}_n^* \times \mathbb{Z}_n$, para romper el criptosistema es

suficiente encontrar dos parejas de mensajes básicos (m_1, c_1) y (m_2, c_2), donde c_i es el mensaje cifrado de m_i, que nos sirvan para resolver el sistema de ecuaciones:

$$\begin{array}{rcrcl} am_1 & + & b & = & c_1 \\ am_2 & + & b & = & c_2 \end{array}$$

en $\mathbb{Z}_n$. Obsérvese que si los vectores (m_1, c_1) y (m_2, c_2) son linealmente independientes, lo cual sucede con alta probabilidad, entonces el sistema tiene solución única.

En el caso en el que $M = C = \mathbb{Z}_n^d$ y $K = \mathrm{GL}_d(\mathbb{Z}_n) \times \mathbb{Z}_n^d$, el análisis es similar, aunque cuanto mayor sea d, más seguro será el criptosistema. En cualquier caso, si se cuenta con una cantidad suficientemente grande de parejas de mensajes básicos en claro con sus correspondientes cifrados, el sistema se puede romper de forma bastante fácil. Más concretamente, supongamos que disponemos de una lista de parejas de mensajes en claro y cifrados $(m_1, c_1), \ldots, (m_k, c_k)$, con $m_k, c_k \in \mathbb{Z}_n^d$, y deseamos calcular la clave (X, y) que se ha utilizado para cifrar con $X \in \mathrm{GL}_d(n)$ e $y \in \mathbb{Z}_n^d$. Planteamos el siguiente sistema de ecuaciones:

$$\left.\begin{array}{rcl} Xm_1 + y & = & c_1 \\ & \vdots & \\ Xm_k + y & = & c_k \end{array}\right\}$$

Desarrollando el sistema se obtiene un sistema de ecuaciones lineales en el que las incógnitas son las entradas de la matrix X y el vector y. Si k es suficientemente grande probablemente el sistema es determinado y podremos encontrar la solución. Más concretamente, escribimos cada m_i como $(m_{i1}, \ldots, m_{id})$ y cada c_i como $(c_{i1}, \ldots, c_{id})$. Escribimos también las incógnitas X e y, que corresponden a la clave buscada, en términos de sus coordenadas:

$$X = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1d} \\ x_{21} & x_{22} & \ldots & x_{2d} \\ \ldots & \ldots & \ldots & \ldots \\ x_{d1} & x_{d2} & \ldots & x_{dd} \end{pmatrix}, \quad y = (y_1, \ldots, y_d)$$

Entonces la ecuación $Xm_i + y = c_i$ se convierte en un sistema de ecuaciones lineales:

$$\begin{array}{c} m_{11}x_{j1} + m_{12}x_{j2} + \cdots + m_{1d}x_{jd} + y_j = c_{1j} \\ m_{21}x_{j1} + m_{22}x_{j2} + \cdots + m_{2d}x_{jd} + y_j = c_{2j} \\ \ldots\ldots\ldots\ldots \\ m_{k1}x_{j1} + m_{k2}x_{j2} + \cdots + m_{kd}x_{jd} + y_j = c_{kj} \end{array}$$

Este sistema es compatible pues los c_{ij} se han calculado a partir de los m_{ij} utilizando estas expresiones para ciertos valores de los x_{ij} e y_i. Si la matriz

$$A = \begin{pmatrix} m_{11} & m_{12} & \ldots & m_{1d} & 1 \\ m_{21} & m_{22} & \ldots & m_{2d} & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ m_{k1} & m_{k2} & \ldots & m_{kd} & 1 \end{pmatrix}$$

tiene rango $d+1$, el sistema tiene una única solución que podemos obtener con un poquito de álgebra lineal. La solución obtenida proporciona la fila j-ésima de la matriz X, y las coordenada j-ésima de y.

Ahora vamos a critoanalizar el Criptosistema de Sustitución que vimos en los Ejemplos 1.1. Supongamos que utilizamos un alfabeto con las letras mayúsculas del castellano y un símbolo para el espacio y el cifrado consiste en hacer una permutación de las letras del alfabeto. Supongamos que el siguiente texto es el resultado de cifrar un texto en castellano con alguna clave que desconocemos.

FMRJG FGZCRYFRWLTECGJRHFYÑXCRLRMCHRWJMJZ
LTFHROÑFRMCR JFTCGRECGRÑGLRWFVEMLRYFRYFH
FCRNRLQJZLEJCGRMLHRFGZLXLHRZLEZJELHRYFRM
LRTLYJCRFTLGRCPJ LHRB FTWJZFRMLRECWÑGJEL
EJCGRYJTFEZLRFGZTFRYCHRBÑGZCHRHJGRGFEFHJ
YLYRYFRÑGRELPMFRFGZTFRMCHRFWBMLVLWJFGZCH

Obsérvese que el número de claves es inmenso: $27! \approx 10^{28}$. Si fuéramos capaces de probar una clave por segundo, tardaríamos uno 10^{20} años en probarlas todas. Por tanto, un ataque por fuerza bruta está descartado. Esto da una apariencia de seguridad que como veremos es falsa pues el sistema es vulnerable con un análisis de frecuencias.

En la primera fila de la siguiente tabla aparecen los símbolos ordenados en frecuencia decreciente en un texto estándar en claro y en la segunda en frecuencia decreciente en el texto dado.

_	E	A	O	N	S	I	R	D	M	L	C	T	P	U	B	F	G	X	Y	Q	H	V	K	J	Z	W	Ñ
R	F	L	G	J	C	H	Z	E	Y	M	T	W	Ñ	_	B	X	V	P	Q	O	N	U	S	K	I	D	A

Es razonable pensar que la permutación utilizada para cifrar se parezca a la de esta tabla, con lo que podríamos conjeturar que la permutación inversa se parezca a la inversa, que es la siguiente.

_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
U	Ñ	B	S	W	D	E	O	I	Z	N	J	A	L	H	P	Q	X	Y	_	K	C	V	G	T	F	M	R

Descifrando con esta clave obtenemos el siguiente texto:

EL NOUEORS ME TACDSON IEMPFS A LSI TNLNRACEI QPE
LS UNECSO DSO POA TEGDLA ME MEIES H AMTNCADNSO LAI
EORAFAI RADRNDAI ME LA CAMNS ECAO SXNAI BECTNRE
LA DSTPONDADNSO MNCEDRA EORCE MSI BPORSI INO
OEDEINMAM ME PO DAXLE EORCE LSI ETBLAGATNEORSI

Obviamente no es el texto original pero al menos las palabras parecen tener una longitud razonable con lo que probablemente el espacio esté bien descifrado. Hemos subrayado una palabra que podría ser "QUE". Por tanto, lo que hacemos ahora es proceder a intercambiar las "P" y las "U".

EL NOPEORS ME TACDSON IEMUFS A LSI TNLNRACEI QUE
LS PNECSO DSO UOA TEGDLA ME MEIES H AMTNCADNSO LAI
PEORAFAI RADRNDAI ME LA CAMNS ECAO SXPNAI BECTNRE
LA DSTUONDADNSO MNCEDRA EORCE MSI BUORSI INO
OEDEINMAM ME UO DAXLE EORCE LSI ETBLAGATNEORSI

La palabra subrayada ahora podría ser una "Y" con lo que intercambiamos las "H" e "Y" para obtener

EL NOPEORS ME TACDSON IEMUFS A LSI TNLNRACEI QUE
LS PNECSO DSO UOA TEGDLA ME MEIES Y AMTNCADNSO LAI
PEORAFAI RADRNDAI ME LA CAMNS ECAO SXPNAI BECTNRE
LA DSTUONDADNSO MNCEDRA EORCE MSI BUORSI INO
OEDEINMAM ME UO DAXLE EORCE LSI ETBLAGATNEORSI

Las palabras subrayadas sugieren ahora convertir las "I" en "S"s y las "S" en "O", con lo que cambiaremos las "O" por "I".

EL NIPEIRO ME TACDOIN SEMUFO A LOS TNLNRACES QUE
LO PNECOI DOI UIA TEGDLA ME MESEO Y AMTNCADNOI LAS
PEIRAFAS RADRNDAS ME LA CAMNO ECAI OXPNAS BECTNRE
LA DOTUINDADNOI MNCEDRA EIRCE MOS BUIROS SNI
IEDESNMAM ME UI DAXLE EIRCE LOS ETBLAGATNEIROS

Ahora parece razonable intercambiar "I" y "N".

EL INPENRO ME TACDONI SEMUFO A LOS TILIRACES QUE
LO PIECON DON UNA TEGDLA ME MESEO Y AMTICADION LAS
PENRAFAS RADRIDAS ME LA CAMIO ECAN OXPIAS BECTIRE
LA DOTUNIDADION MICEDRA ENRCE MOS BUNROS SIN
NEDESIMAM ME UN DAXLE ENRCE LOS ETBLAGATIENROS

¿Será que las "M" y "D" están intercambiadas?

EL INPENRO DE TACMONI SEDUFO A LOS TILIRACES QUE
LO PIECON MON UNA TEGMLA DE DESEO Y ADTICAMION LAS
PENRAFAS RAMRIMAS DE LA CADIO ECAN OXPIAS BECTIRE
LA MOTUNIMAMION DICEMRA ENRCE DOS BUNROS SIN
NEMESIDAD DE UN MAXLE ENRCE LOS ETBLAGATIENROS

En realidad la "M", que antes era "D", parece que debe ser una "C" y la "F" parece que debe ser "J". Por tanto, intercambiamos las dos primeras entre sí y lo mismo hacemos con las dos últimas.

EL INPENRO DE TAMCONI SEDUJO A LOS TILIRAMES QUE
LO PIEMON CON UNA TEGCLA DE DESEO Y ADTIMACION LAS
PENRAJAS RACRICAS DE LA MADIO EMAN OXPIAS BEMTIRE
LA COTUNICACION DIMECRA ENRME DOS BUNROS SIN
NECESIDAD DE UN CAXLE ENRME LOS ETBLAGATIENROS

¿Tácticas de la radio eran?: "M" → "R" → "T".

> EL INPENTO DE MARCONI SEDUJO A LOS MILITARES QUE
> LO PIERON CON UNA MEGCLA DE DESEO Y ADMIRACION LAS
> PENTAJAS TACTICAS DE LA RADIO ERAN OXPIAS BERMITE
> LA COMUNICACION DIRECTA ENTRE DOS BUNTOS SIN
> NECESIDAD DE UN CAXLE ENTRE LOS EMBLAGAMIENTOS

"X" → "B" → "P" → "V" → "X".

> EL INVENTO DE MARCONI SEDUJO A LOS MILITARES QUE
> LO VIERON CON UNA MEZCLA DE DESEO Y ADMIRACION LAS
> VENTAJAS TACTICAS DE LA RADIO ERAN OBVIAS PERMITE
> LA COMUNICACION DIRECTA ENTRE DOS PUNTOS SIN
> NECESIDAD DE UN CABLE ENTRE LOS EMPLAZAMIENTOS.

Este ejemplo muestra que una permutación de las letras no es una idea adecuada porque la frecuencia de los símbolos en los mensajes en claro y cifrados es similar, aunque en otro orden. Esto permite un análisis de frecuencias. La moraleja es que el conjunto de mensajes básicos no puede ser pequeño y la frecuencia con la que aparecen en el texto debe ser lo más uniforme posible para que no sea fácil obtener información parcial del mensaje. Como hemos visto el uso del espacio ha introducido en el mensaje unas pistas que nos han ayudado a identificar donde acaba y termina cada palabra. Aunque esta información parcial pueda parecer nimia, hemos visto que no lo es. Una táctica para aumentar el número de símbolos es utilizar bloques de símbolos. Por ejemplo, si usamos el alfabeto español, en lugar de considerar cada letra como un mensaje básico consideramos listas de una longitud fijada como los mensajes básicos que, como explicamos en varios ejemplos anteriores, los convertimos en objetos matemáticos (por ejemplo números) para proceder a encriptarlos. Esto no solo aumenta el conjunto de mensajes básicos sino también homogeniza su distribución de frecuencias en un texto largo. El concepto de entropía que vemos en la siguiente sección trata esta idea de forma más rigurosa.

1.3 Entropía

El objetivo de esta sección es introducir el concepto de entropía. La entropía de una variable aleatoria discreta es la incertidumbre sobre el suceso que va a ocurrir si elegimos un suceso al azar. Visto desde otro punto de vista, podríamos decir que la entropía es una medida de la cantidad de información que ganamos al elegir un suceso al azar. Por ejemplo, supongamos que uno de los sucesos tiene probabilidad 1, y por tanto todos los demás tienen probabilidad 0. Esta claro que en este caso la incertidumbre es nula y, por tanto, la información obtenida al elegir un suceso al azar es nula. En el extremo opuesto, supongamos que todos los sucesos tienen la misma probabilidad. En tal caso, la incertidumbre es máxima, o equivalentemente, la cantidad de información ganada al elegir un suceso al azar es máxima. Todas las variables aleatorias que nos interesan son discretas y de hecho tienen un conjunto de sucesos finito. Por tanto nos restringiremos a ese caso.

Una *distribución de probabilidad finita* es una tupla $(p_1, \ldots, p_n)$ de números reales no negativos que suman 1. Una *variable aleatoria* con espacio de sucesos finito (o *variable aleatoria*

discreta) es una aplicación $X : S \to [0,1]$ donde S es un conjunto finito, llamado el *espacio de sucesos* y $(p(s))_{s\in S}$ es una función de probabilidad finita.

Esta claro que el concepto de entropía que estamos intentado buscar no debe depender de la naturaleza de los sucesos sino exclusivamente de la distribución de probabilidad asociada, por tanto consideraremos la entropía de una distribución de probabilidad finita como la entropía de una variable aleatoria con dicha función de distribución de probabilidad en un cierto orden de los sucesos. En resumen estamos buscando una función real que está definida sobre las posibles distribuciones de probabilidad, o sea una función H definida sobre el conjunto de las sucesiones finitas $(p_1, \ldots, p_n)$ de números reales no negativos que sumen 1. Veamos algunas de las propiedades que deseamos para esta función:

- **Continuidad**. La restricción de la función de entropía a las funciones de distribución de una longitud prefijada debe ser continua porque pequeñas variaciones en la distribución de probabilidades no deben suponer grandes cambios en la incertidumbre.

- **Creciente respecto del número de sucesos**. En las distribuciones uniformes la incertidumbre crece con el número de sucesos. Por tanto la función de entropía debe cumplir
$$H\left(\frac{1}{n}, \ldots, \frac{1}{n}\right) < H\left(\frac{1}{n+1}, \ldots, \frac{1}{n+1}\right).$$

- **Independiente de agrupamientos**. Sea X una variable aleatoria con espacio de sucesos S. Supongamos que $S' = \{S_1, \ldots, S_m\}$ es una partición del conjunto de sucesos S. Sea X' la variable aleatoria con espacio de sucesos S' de forma que $p(X' = S_i) = P(X \in S_i)$. Para cada $i = 1, \ldots, m$ sea X_i la variable aleatoria con espacio de sucesos S_i de forma que $P(X_i = s)$ es la probabilidad de que $X = s$, condicionado a que $X' = S_i$, o sea $P(X_i = s) = P(X = ss \mid X' = S_i)$. Esta claro que obtenemos la misma cantidad de información eligiendo un suceso de X' y, en el caso en que $X' = S_i$, eligiendo un suceso de X_i, que eligiendo directamente un suceso de X. Por tanto vamos a exigir que la entropía de X sea la suma de la entropía de X' más la media de las entropías de los X_i ponderada por las probabilidades de los sucesos $X' = S_i$. En particular, si X tiene la distribución uniforme con n sucesos entonces $P(X = s) = 1/n$, para todo $s \in S$ y si S_i tiene k_i elementos, entonces la probabilidad del suceso $P(X' = S_i = \frac{k_i}{n}$ y $P(X_i = s) = \frac{1}{k_i}$. Entonces la entropía de X es $H(\frac{1}{n}, \ldots, \frac{1}{n})$, la entropía de X' es $H(\frac{k_1}{n}, \ldots, \frac{k_m}{n})$ y la entropía de F_i es $H(\frac{1}{k_i}, \ldots, \frac{1}{k_i})$. Por tanto, otra de las propiedades deseadas es la siguiente:
$$H\left(\frac{1}{n}, \ldots, \frac{1}{n}\right) = H\left(\frac{k_1}{n}, \ldots, \frac{k_m}{n}\right) + \sum_{i=1}^{m} \frac{k_i}{n} H\left(\frac{1}{k_i}, \ldots, \frac{1}{k_i}\right),$$
siempre que $\sum_{i=1}^{m} k_i = n$.

Sorprendentemente la función de entropía está "prácticamente" unívocamente determinada por las propiedades anteriores.

Teorema 1.2 *Una función continua definida sobre el conjunto de funciones de distribución de longitud n que cumpla las siguientes condiciones:*

(1) $H(\frac{1}{n},\dots,\frac{1}{n}) < H(\frac{1}{n+1},\dots,\frac{1}{n+1})$.

(2) $H(\frac{1}{n},\dots,\frac{1}{n}) = H(\frac{k_1}{n},\dots\frac{k_m}{n}) + \sum_{i=1}^{m}\frac{k_i}{n}H(\frac{1}{k_i},\dots,\frac{1}{k_i})$ *siempre que* $\sum_{i=1}^{m} k_i = n$.

es de la forma

$$H(p_1,\dots,p_n) = \sum_{i=1}^{n} p_i \log_b \frac{1}{p_i} = -\sum_{i=1}^{n} p_i \log_b p_i$$

para algún $b > 1$. *Donde* $0\log_b 0$ *es interpretado como* 0.

Demostración. Para cada divisor m de n se tiene que verificar

$$\begin{array}{rl} H\left(\frac{1}{n},\dots,\frac{1}{n}\right) & = H\left(\frac{m}{n},\dots,\frac{m}{n}\right) + \sum_{i=1}^{\frac{n}{m}}\frac{m}{n}H\left(\frac{1}{m},\dots,\frac{1}{m}\right) \\ & = H\left(\frac{m}{n},\dots,\frac{m}{n}\right) + H\left(\frac{1}{m},\dots,\frac{1}{m}\right) \end{array}$$

En particular, si $n = m^s$, entonces

$$H\left(\frac{1}{m^s},\dots,\frac{1}{m^s}\right) = H\left(\frac{1}{m^{s-1}},\dots,\frac{1}{m^{s-1}}\right) + H\left(\frac{1}{m},\dots,\frac{1}{m}\right).$$

Si ponemos $g(n) = H\left(\frac{1}{n},\dots,\frac{1}{n}\right)$, entonces hemos visto que

$$g(m^s) = g(m^{s-1}) + g(m),$$

y, por inducción sobre s se obtiene que

$$g(m^s) = sg(m).$$

La condición (1) implica que g es estrictamente creciente y, por tanto, para todo $m > 1$, tenemos $g(m^s) < g(m^{s+1})$, o lo que es igual $sg(m) \le (s+1)g(m)$. Por tanto $g(m)$ es positivo.

Sean n, k y $m \neq 1$ enteros y sea s otro entero tal que $m^s \le n^k < m^{s+1}$. Como g es estrictamente creciente $g(m^s) \le g(n^k) < g(m^{s+1})$, o equivalentemente

$$sg(m) \le kg(n) < (s+1)g(m).$$

Como log también es una función creciente también tenemos

$$s\log(m) \le k\log(n) < (s+1)\log(m).$$

Por tanto

$$\frac{s}{k} \le \frac{g(n)}{g(m)}, \frac{\log(n)}{\log(m)} < \frac{s+1}{k}.$$

Luego

$$-\frac{1}{k} \le \frac{g(n)}{g(m)} - \frac{\log(n)}{\log(m)} < \frac{1}{k}.$$

Como k es arbitrario

$$\frac{g(n)}{g(m)} = \frac{\log(n)}{\log(m)},$$

o sea,

$$\frac{g(n)}{\log(n)} = \frac{g(m)}{\log(m)} = C$$

Luego $g(n) = C \log n$ para algún número positivo n. Por tanto, si elegimos C de forma adecuada tendremos que $g(n) = \log_b n$.

Supongamos ahora que $(p_1, \ldots, p_k)$ es una distribución de probabilidad formada por números racionales. Poniéndolos con común denominador podemos suponer que $p_i = \frac{b_i}{n}$ y, de la propiedad (2) tenemos

$$\begin{aligned} H(p_1, \ldots, p_k) &= H\left(\frac{b_1}{n}, \ldots, \frac{b_k}{n}\right) = g(n) - \sum_{i=1}^{k} \frac{b_i}{n} g(b_i) = \log_b n - \sum_{i=1}^{k} \frac{b_i}{n} \log_b b_i \\ &= \sum_{i=1}^{k} \frac{b_i}{n} \log_b \frac{n}{b_i} = \sum_{i=1}^{k} p_i \log_b \frac{1}{p_i}. \end{aligned}$$

Como H es continua, entonces

$$H(p_1, \ldots, p_k) = \sum_{i=1}^{k} p_i \log_b \frac{1}{p_i}$$

para toda k-upla $(p_1, \ldots, p_k)$ de números reales en el dominio de H. ∎

Definición 1.3 *Sea b un número real mayor que 1. Se llama* entropía *en base b de una distribución de probabilidad $P = (p_1, \ldots, p_k)$ a*

$$H_b(p_1, \ldots, p_k) = \sum_{i=1}^{k} p_i \log_b \frac{1}{p_i}.$$

La entropía de una variable aleatoria discreta es la entropía de su distribución de probabilidad.

La base b en la que se calcule la función de entropía sólo implica un cambio de escala debido a la igualdad $\log_b x = \log_{b'} x \cdot \log_b b'$ que implica que

$$H_b(X) = H_{b'}(X) \cdot \log_b b'.$$

En consecuencia, a menudo omitiremos la base y simplemente escribiremos $H(X)$ ó $H(p_1, \ldots, p_k)$.

Vamos a cerrar esta sección con una generalización de la propiedad (2) del Teorema 1.2.

Proposición 1.4 *Sea $(p_1, \ldots, p_n, q_1, \ldots, q_m)$ una distribución de probabilidad. Si $a = \sum_{i=1}^{n} p_i$, entonces*

$$H(p_1, \ldots, p_n, q_1, \ldots, q_m) = H(a, 1-a) + aH\left(\frac{p_1}{a}, \ldots, \frac{p_n}{a}\right) + (1-a)\, H\left(\frac{q_1}{1-a}, \ldots, \frac{q_m}{1-a}\right).$$

Demostración.

$$\begin{aligned}
&H\left(p_1,\ldots,p_n,q_1,\ldots,q_m\right)\\
&=\textstyle\sum_{i=1}^n p_i\log\frac{1}{p_i}+\sum_{i=1}^m q_i\log\frac{1}{q_i}\\
&=\textstyle\sum_{i=1}^n p_i\log\frac{a}{ap_i}+\sum_{i=1}^m q_i\log\frac{1-a}{(1-a)q_i}\\
&=\textstyle\sum_{i=1}^n p_i\left(\log\frac{a}{p_i}+\log\frac{1}{a}\right)+\sum_{i=1}^m q_i\left(\log\frac{1-a}{q_i}+\log\frac{1}{1-a}\right)\\
&=\textstyle\sum_{i=1}^n p_i\log\frac{1}{a}+\sum_{i=1}^n p_i\log\frac{a}{p_i}+\sum_{i=1}^m q_i\log\frac{1}{1-a}+\sum_{i=1}^m q_i\log\frac{1-a}{q_i}\\
&=\log\tfrac{1}{a}\textstyle\sum_{i=1}^n p_i+\log\frac{1}{1-a}\sum_{i=1}^m q_i+a\sum_{i=1}^n\frac{p_i}{a}\log\frac{a}{ap_i}+(1-a)\sum_{i=1}^m\frac{q_i}{1-a}\log\frac{1-a}{q_i}\\
&=a\log\tfrac{1}{a}+(1-a)\log\tfrac{1}{1-a}+a\textstyle\sum_{i=1}^n\frac{p_i}{a}\log\frac{a}{ap_i}+(1-a)\sum_{i=1}^m\frac{q_i}{1-a}\log\frac{1-a}{q_i}\\
&=H\left(a,1-a\right)+aH\left(\tfrac{p_1}{a},\ldots,\tfrac{p_n}{a}\right)+(1-a)\,H\left(\tfrac{q_1}{1-a},\ldots,\tfrac{q_m}{1-a}\right).
\end{aligned}$$

∎

Vamos ahora a ver cual es el rango de la función de entropía. Más concretamente vamos a demostrar el siguiente.

Teorema 1.5 *Sea X una variable aleatoria discreta con n sucesos posibles. Entonces*

$$0\le H_b(X)\le\log_b n.$$

Además, $H(X)=0$ precisamente si $P(X=x)=1$ para algún suceso x y $H_b(X)=\log_b n$ si y sólo si la distribución de frecuencias de X es uniforme.

Para demostrar el Teorema 1.5 necesitaremos dos lemas. El primero es bien conocido:

Lema 1.6 *Para todo número real positivo x se verifica $\ln x\le x-1$ y la igualdad se verifica precisamente si $x=1$.*

El segundo es un poco más complicado:

Lema 1.7 *Sea $P=(p_1,\ldots,p_n)$ una distribución de probabilidad y $Q=(q_1,\ldots,q_n)$ una sucesión de números reales tales que $0\le q_i\le 1$ y $\sum_{i=1}^n q_i\le 1$. Entonces*

$$\sum_{i=1}^n p_i\log\frac{1}{p_i}\le\sum_{i=1}^n p_i\log\frac{1}{q_i}$$

(donde se interpreta $0\log\frac{1}{0}=0$ y $p\log\frac{1}{0}=+\infty$, si $p>0$). Además la igualdad se verifica precisamente si $p_i=q_i$ para todo i.

Demostración. En primer lugar, podemos suponer que los logaritmos son logaritmos naturales, ya que las expresiones que aparecen a ambos lados de la desigualdad varían según un factor positivo para diferentes bases. Obsérvese que por los convenios establecidos

$$\text{si } pq=0 \text{ entonces } p\ln p\le p\ln q+q-p.$$

Del Lema 1.6 se tiene que en caso que $pq \neq 0$ se verifica que

$$\ln \frac{q}{p} \leq \frac{q}{p} - 1$$

y, por tanto

$$p \ln \frac{1}{p} \leq p \ln \frac{1}{q} + q - p.$$

En consecuencia

$$\sum_{i=1}^{n} p_i \log \frac{1}{p_i} \leq \sum_{i=1}^{n} p_i \left(\log \frac{1}{q_i} + q_i - p_i \right) = \sum_{i=1}^{n} p_i \log \frac{1}{q_i} + \sum_{i=1}^{n} p_i (q_i - p_i) \leq \sum_{i=1}^{n} p_i \log \frac{1}{q_i}.$$

y la igualdad entre los dos extremos se verifica precisamente si existe igualdad en cada uno de los pasos intermedios. Para que exista igualdad en el primer paso hace falta que

$$p_i \ln \frac{1}{p_i} = p_i \ln \frac{1}{q_i} + q_i - p_i$$

para todo i, o equivalentemente que

$$\ln \frac{q_i}{p_i} = \frac{q_i}{p_i} - 1.$$

Pero del Lema 1.6 se tiene que eso equivale a que $q_i = p_i$ para todo i. ∎

Demostración del Teorema 1.5. Supongamos que $(p_1, \ldots, p_n)$ es la distribución de probabilidad de la variable aleatoria X. Sea $q_i = \frac{1}{n}$. Del Lema 8 deducimos que

$$H(X) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i} \leq \sum_{i=1}^{n} p_i \log \frac{1}{1/n} = \log n \sum_{i=1}^{n} p_i = \log n$$

y que la igualdad se verifica precisamente si $p_i = \frac{1}{n}$ para todo i. ∎

A menudo, para una variable aleatoria en la que el espacio de sucesos tiene cardinal k se suele elegir base k. En tal caso obsérvese que la distribución uniforme tiene entropía 1. En efecto,

$$H_k \left(\frac{1}{k}, \ldots, \frac{1}{k} \right) = \sum_{i=1}^{k} \frac{1}{k} \log_k k = 1.$$

O sea, solemos considerar como unidad de entropía a la entropía de la distribución uniforme. Pero hay que tener cuidado con esta consideración ya que estaría en contra de la condición que habíamos exigido para la función de entropía sobre las distribuciones uniformes:

$$H \left(\frac{1}{n}, \ldots, \frac{1}{n} \right) < H \left(\frac{1}{n+1}, \ldots, \frac{1}{n+1} \right).$$

O sea, la unidad de entropía depende de la base en la que decidamos calcular el logaritmo. Esa elección la haremos según el entorno en el que nos movamos y, cuando estemos tratando

de variables aleatorias con un espacio de sucesos de cardinal fijado, habitualmente elegiremos dicho cardinal como base de logaritmos.

La tabla siguiente muestra la distribución de frecuencias de las 27 letras del alfabeto español[1]:

A	B	C	D	E	F	G	H	I
0.1251	0.014	0.0466	0.0584	0.1366	0.0068	0.0099	0.0068	0.0623
J	K	L	M	N	Ñ	O	P	Q
0.0043	0.0001	0.0495	0.0313	0.0669	0.003	0.0866	0.0249	0.0086
R	S	T	U	V	W	X	Y	Z
0.0685	0.0796	0.0461	0.0391	0.0088	0.0002	0.0021	0.0088	0.0051

La entropía de esta distribución de frecuencia, calculada con logaritmos en base 27 para obtener un valor en el intervalo $[0, 1]$, es aproximadamente 0.85. Para comprobar esto vamos a usar la siguiente implementación para GAP de la función de entropía.

```
Entropia := function(x)
local y;
y := Filtered(x,u->u>0);
return -Sum(y,u->u*Log(u)/Log(Float(Length(x))));
end;
```

Usamos esta función para calcular la entropía de la distribución de frecuencia de las letras en castellano.

```
gap> FreqAlfaEsp :=
> [ 0.1251, 0.014, 0.0466, 0.0584, 0.1366, 0.0068, 0.0099, 0.0068,
> 0.0623, 0.0043, 0.0001, 0.0495, 0.0313, 0.0669, 0.003, 0.0866,
> 0.0249, 0.0086, 0.0685, 0.0796, 0.0461, 0.0391, 0.0088, 0.0002,
> 0.0021, 0.0088, 0.0051 ];;
gap> Entropia(FreqAlfaEsp);
0.846582
```

Por tanto la distribución de frecuencias relativas de un texto en claro en español suficientemente largo escrito sólo con las 27 letras del alfabeto (o sea no se usan ni números, ni espacios, ni signos de puntuación) deberá aproximarse a 0.85. Un texto cifrado con el Criptosistema de César tendrá la misma distribución de frecuencias que un texto en claro, salvo una permutación, y por tanto tendrá la misma entropía. Sin embargo un texto cifrado con el criptosistema de Vigenère tendrá una distribución de frecuencias más parecida a la distribución uniforme y por tanto su entropía deberá de ser mayor que 0.85. Si supiéramos la longitud de una clave Vigenère, podríamos aplicar el mismo análisis de frecuencias que el utilizado para romper el Criptosistema de César. Supongamos que la longitud de la clave es m y dividimos el texto en m subtextos formados por los símbolos situados a distancia m. Cada uno de estos subtextos será como un texto cifrado con el Criptosistema de César y en consecuencia tendrá la

[1]Fuente: http://es.wikipedia.org/wiki/Frecuencia_de_aparición_de_letras. La hemos modificado ligeramente para que la suma de las probabilidades sea realmente 1

misma entropía que un texto en claro. Por tanto, podemos calcular la entropía media de estos m textos para diferentes valores de m con la esperanza de encontrar un valor de m para el que la entropía media de los subtextos coincida con la de un texto en claro estándar. Este sistema funciona incluso si no conocemos la entropía del alfabeto original pues simplemente podemos ir calculando la entropía hasta que encontremos un valor de m para el que la entropía disminuya significativamente. En la siguiente sección veremos cómo hacer esto de forma práctica.

1.4 Introducción a GAP y criptoanálisis de algunos criptosistemas

En esta sección vamos usar el concepto de entropía para con ayuda del programa GAP descifrar algunos mensajes.

En primer lugar vamos a adaptar nuestro alfabeto a los 256 caracteres que reconoce GAP. Más concretamente, GAP tiene dos funciones `IntChar` y `CharInt`, mutuamente inversas. La segunda convierte un entero entre 0 y 255 en un carácter y, por supuesto, la segunda hace lo contrario:

```
gap> CharInt(80);
'P'
gap> CharInt(100);
'd'
```

De esta manera podemos convertir un texto en una lista de números:

```
gap> texto:="Yo soy un pobre texto lleno de letras que voy a ser convertido
en guarismos. Vengan por favor a salvarme";
"Yo soy un pobre texto lleno de letras que voy a ser convertido
en guarismos. Vengan por favor a salvarme"
gap> Ntexto:=List(texto,IntChar);
[ 89, 111, 32, 115, 111, 121, 32, 117, 110, 32, 112, 111, 98, 114, 101, 32,
116, 101, 120, 116, 111, 32, 108, 108, 101, 110, 111, 32, 100, 101, 32,
108, 101, 116, 114, 97, 115, 32, 113, 117, 101, 32, 118, 111, 121, 32,
97, 32, 115, 101, 114, 32, 99, 111, 110, 118, 101, 114, 116, 105, 100, 111, 32,
101, 110, 32, 103, 117, 97, 114, 105, 115, 109, 111, 115, 46, 32,
86, 101, 110, 103, 97, 110, 32, 112, 111, 114, 32, 102, 97, 118, 111, 114, 32,
97, 32, 115, 97, 108, 118, 97, 114, 109, 101 ]
gap> List(Ntexto,CharInt);
"Yo soy un pobre texto lleno de letras que voy a ser convertido
en guarismos. Vengan por favor a salvarme"
```

Hemos evitado escribir acentos, eñes y otros símbolos no porque sea posible sino porque para hacerlo hay que escribirlos en un documento aparte y leerlo. Por ejemplo podemos escribir lo siguiente en un documento de texto:

```
texto := "La construcción ‘tratarse de’ con este significado de ‘ser’
```

```
se usa para referirnos a algo anteriormente mencionado, señala
el Diccionario panhispánico de dudas, de la Real Academia Española.
Leamos este texto: ‘Llegaron por la mañana a Vilaflor.
Se trata de un pueblo tranquilo situado a los pies de un inmenso volcán’. ";
```

Por ejemplo yo lo he escrito en un documento que he llamado “Texto.gap” y lo he guardado en la carpeta /Users/ANGEL/Documents/GAP/Criptografia/. Después lo he leído con el siguiente comando en GAP:

```
gap> Read("/Users/ANGEL/Documents/GAP/Criptografia/Texto.gap");
```

con lo que puedo ahora usar la variable “texto”.

```
gap> texto;
"La construcción ‘tratarse de’ con este significado de ‘ser’ se usa \
para referirnos a algo anteriormente mencionado, señala el Diccionario panhi\
spánico de dudas, de la Real Academia Española. Leamos este texto: ‘Llega\
ron por la mañana a Vilaflor. Se trata de un pueblo tranquilo situado a los \
pies de un inmenso volcán’."
gap> List(texto,IntChar);
[ 76, 97, 32, 99, 111, 110, 115, 116, 114, 117, 99, 99, 105, 195, 179, 110,
....
108, 99, 195, 161, 110, 226, 128, 153, 46 ]
```

Por supuesto que no hemos puesto la lista completa de números que proporciona el último comando. Podríamos ahora manipular estos números para encriptar el mensaje original con cualquiera de los criptosistemas que hemos visto.

Por ejemplo, ahora vamos hacer una implementación del criptosistema de César por bloques y después vamos a ver como romperlo. Comenzamos construyendo una función de GAP que agrupe los símbolos de un texto en bloques de una determinada longitud y los convierta en sus correspondientes numéricos.

```
Bloques := function(texto,l)
local Ntexto,n,r,q,cf,i;
Ntexto := List(texto,IntChar);
n := Length(Ntexto);
r := n mod l;
q := Int(n/l);
cf := [];
for i in [0..q-1] do
Add( cf , Sum( List( [1..l], x -> Ntexto[x+i*l]*256^(l-x) ) ) );
od;
if r > 0 then
Add( cf , Sum( List( [1..r], x -> Ntexto[x+q*l]*256^(l-x) ) ) );
fi;
return cf;
end;
```

Por ejemplo, si $l = 4$ y `texto = "Un texto para ser troceado en bloques de longitud 4"` entonces la función `Bloques` primero divide `texto` en palabras de longitud 4:

```
"Un t" "exto" "par" "a se" "r tr" "ocea" "do e"
"n bl" "oque" "s de" "lon" "gitu" "d 4"
```

Cada una de estas palabras se convierte en una lista de números usando el comando `IntChar`. Por ejemplo, la primera palabra corresponde a la lista `[ 85, 110, 32, 116 ]`.

```
gap> List("Un t",IntChar);
[ 85, 110, 32, 116 ]
```

Después convertimos cada una de estas listas de números en elementos de $\mathbb{Z}_{256^4}$. Por ejemplo la última lista corresponde a

$$85 \cdot 256^3 + 110 \cdot 256^2 + 32 \cdot 256 + 116 = 1433280628.$$

```
gap> Bloques("Un texto para ser troceado en bloques de longitud 4",4);
[ 1433280628, 1702392943, 544235890, 1629516645, 1914729586, 1868784993,
1685004389, 1847616108, 1869706597, 1931502693, 543977326, 1734964341,
1679832064 ]
```

Si l es la longitud de los bloques entonces para cifrar un mensaje lo convertimos en una lista de números usando `Bloques` y le sumamos en $\mathbb{Z}_{256^l}$ a cada uno de los números la clave que tendrá que ser un número entre 0 y $256^l - 1$.

```
Cesar := function(texto,clave,l)
local BloquesTexto;
BloquesTexto := Bloques(texto,l);
return List(BloquesTexto,x-> (x+clave) mod 256^l );
end;
```

Para descifrar usamos la función `DesCesar` que usa a su vez la función `CB` que lo que hace es escribir un número en $\mathbb{Z}_{256^l}$ en su expresión es base 256.

```
CB := function(n,l)
local q,out,i,r;
if n >= 256^l then
return fail;
fi;
q := n;
out := [];
for i in [1..l] do
r := q mod 256;
Add(out,r,1);
q := (q-r)/256;
od;
```

```
return out;
end;
#########################################################
DesCesar := function(cf,clave,l)
local BloquesTexto,Ntexto,x;
BloquesTexto := List(cf,x-> (x-clave) mod 256^l );
Ntexto := [];
for x in BloquesTexto do
Ntexto := Concatenation(Ntexto,CB(x,l));
od;
return List(Ntexto,CharInt);
end;

gap> cf := Cesar("Un texto para ser troceado en bloques de longitud 4",10^4,4);
[ 1433290628, 1702402943, 544245890, 1629526645, 1914739586, 1868794993,
1685014389, 1847626108, 1869716597, 1931512693, 543987326, 1734974341,
1679842064 ]
gap> DesCesar(cf,10^4,4);
"Un texto para ser troceado en bloques de longitud 4\000"
```

El símbolo `\000` que se ha añadido al final es porque el último bloque tiene longitud menor que 4.

Ahora vamos a ver cómo podemos romper el criptosistema de César por bloques usando que el espacio, o sea `' '` suele ser mucho más común que cualquier otro símbolo.

Supongamos que el texto cifrado es la lista de números cf y que la clave desconocida es $c \in \mathbb{Z}_{256,}$. Sea $[c_0, c_1, \ldots, c_{l-1}]$ la expresión de c en base 256, o sea $c = c_0 + c_1 256 + c_2 256^2 + \cdots + c_{l-1} 256^{l-1}$. Lo que vamos a hacer es descubrir uno a uno los valores de c_i. Lo ilustramos con un texto cifrado con $l = 5$:

```
gap> cf := Cesar(Pactos,clave,5);
[ 280938180612, 139221282305, 435591654922, 473088162825, 435808927990,
...491465128694, 494563283254, 494563263233, 500083426041, 452921425918,
418642202005 ]
```

Solo hemos escrito los primeros y últimos números pero la longitud de `cf` es 611. Para descubrir c_0 calculamos la lista cf_0 obtenida reduciendo los elementos de cf módulo 256 y buscamos en el cf_0 el número más repetido. Obsérvese que al cifrar se ha sumado c, con lo cual a cada uno de las los valores por `IntChar` de los símbolos del texto original que estaban en las posiciones múltiplo de l se les ha sumado c_0 módulo 256 a su correspondiente numérico. Por tanto, el valor más repetido corresponderá probablemente al espacio.

```
gap> cf0 := List(cf,x-> x mod 256);;
gap> sim := SSortedList(cf0);
[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 54, 66, 72, 79, 88, 149, 181, 186, 188,
193, 195, 196, 197, 198, 199, 200, 201, 202, 204, 205, 207, 215, 216, 226,
```

```
228, 229, 230, 231, 232, 233, 234, 235, 246, 247, 248, 249, 250, 251, 252,
253, 254 ]
gap> List(sim,x->[x,Length(Positions(cf0,x))]);
[ [ 1, 24 ], [ 2, 12 ], [ 3, 36 ], [ 4, 43 ], [ 5, 11 ], [ 6, 9 ], [ 7, 34 ],
[ 8, 37 ], [ 9, 27 ], [ 10, 22 ], [ 14, 2 ], [ 54, 2 ], [ 66, 4 ],
[ 72, 4 ], [ 79, 2 ], [ 88, 6 ], [ 149, 1 ], [ 181, 100 ], [ 186, 3 ],
[ 188, 1 ], [ 193, 6 ], [ 195, 5 ], [ 196, 1 ], [ 197, 1 ], [ 198, 3 ],
[ 199, 1 ], [ 200, 2 ], [ 201, 3 ], [ 202, 2 ], [ 204, 2 ], [ 205, 1 ],
[ 207, 1 ], [ 215, 2 ], [ 216, 3 ], [ 226, 1 ], [ 228, 2 ], [ 229, 8 ],
[ 230, 1 ], [ 231, 1 ], [ 232, 3 ], [ 233, 1 ], [ 234, 2 ], [ 235, 1 ],
[ 246, 34 ], [ 247, 5 ], [ 248, 20 ], [ 249, 21 ], [ 250, 56 ], [ 251, 2 ],
[ 252, 7 ], [ 253, 3 ], [ 254, 30 ] ]
```

Observamos que el número más repetido en `cf0` es 181 que se repite 100 veces mientras que el siguiente en frecuencia se repita 56 veces. Suponemos que corresponde a haber cifrado espacios. Como el valor por `IntChar` del espacio es 32, asignamos el valor $c_0 = 181 - 32 = 149$.

Para descubrir c_1 hacemos lo mismo pero ahora en lugar de trabajar con cf, trabajamos con la lista formada por sus cocientes al dividir por 256. Observese que las entradas de esta nueva lista son las de $(cf - cf_0)/256$. Para seguir descubriendo los demás valores de los cf_i repetimos lo mismo pero cambiando la lista cf por los cocientes de dividir las entradas de cf por por 256^i.

He aquí la implementación de la función para descubrir la clave en el caso de un texto dividido en bloques de longitud l.

```
################################################

ClaveCesar := function(cf,l)
local CF,clave,i,cfp,sim,cont,x,cs,c;
CF := cf;
clave := 0;
for i in [0..l-1] do
cfp := List(CF,x-> x mod 256 );
sim := SSortedList(cfp);
cont := 0;
for x in sim do
cs := Length(Positions(cfp,x));
if cont < cs then
c := x;
cont := cs;
fi;
od;
clave := clave + (c-32)*256^i;
CF := (CF-cfp)/256;
od;
return clave;
```

```
end;
```

Apliquemos esta función al mensaje cf y observemos la primera parte del mensaje descifrado:

```
gap> cl := ClaveCesar(cf,5);
114757013
gap> des := DesCesar(cf,cl,5);;
gap> List([1..50],x->des[x]);
"Ababo ce ledr un\037inteqesanse arsícuko de\037Adoleo Q"
```

Está claro que no hemos encontrado la clave exacta, pero algo de luz ha aparecido pues reconocemos `inteqesanse` como `interesante` o `arsícuko` como `artículo`. Probablemente uno de los c_i está mal. Si dividimos en bloques de cinco símbolos tendremos las siguientes palabras

```
"Ababo" " ce l" "edr u" "n\037int" "eqesa" "nse a" "rsíc" "uko d" "e\037Ado" "leo Q"
```

Aunque parezca que `rsíc` tiene longitud 4, en realidad tiene longitud 5 pues las letras con acentos tienen longitud 2. De la misma manera, `\037` representa un símbolo único. Observamos que la segunda letra de cada palabra es la que parece estar mal. Por ejemplo `Ababo` podría ser `Acabo` y `"n\037int" "eqesa" "nse a" "rsíc" "uko d"` puede ser `"n int" "eresa" "nte a" "rtíc" "ulo d"`. Comparamos la diferencia de los valores numéricos de las letras que aparecen por lo que suponemos que debería aparecer:

```
gap> IntChar('b')-IntChar('c');
-1
gap> IntChar('\037')-IntChar(' ');
-1
gap> IntChar('q')-IntChar('r');
-1
gap> IntChar('r')-IntChar('s');
-1
```

Por tanto, el valor de c_3 debería ser uno menos, o lo que es lo mismo hay que restar 256^3 a la clave:

```
gap> cl := cl-256^3;
97979797
gap> DesCesar(cf,cl,5);
"Acabo de leer un interesante artículo de Adolfo Quirós, catedrático de ál\
gebra de la UAM en el que analiza matemáticamente la cuota de poder de los pa\
rtidos con representación en el congreso en función del número de coalicion\
es en las que cada partido es imprescindible. Para comprenderlo vamos a consid\
erar un ejemplo sencillo. Supongamos que en el congreso sólo hubiera tres par\
tidos dos que tuvieran 174 diputados y un tercero con 2 diputados. Como son ne\
cesarios 176 votos para elegir presidente hay cuatro pactos ganadores: Uno con\
los tres partidos, o cualquier pacto entre dos. Obsérvese que para formar go\
```

```
bierno, cada uno de los tres partidos tiene exáctamente el mismo poder. Vamos\
con el congreso de diputados real. Para hacer los cálculos se puede utilizar\
el siguiente enlace: https://math.temple.edu/~conrad/cgi-bin/PowerIndex.py. D\
espués de poner la distribución de diputados en Weight, 1 en Multiplicity, 1\
76 en quota y clickar en Submit obtendremos una serie de datos. Sólo nos impo\
rta la columna BPI%, que indica el porcentaje de poder de cada partido. Si lo \
hacemos con el supuesto anterior, es decir, dos partidos con 174 diputados y u\
no con 2 diputados, obtenemos que cada partido tiene un 33,33% de poder, como \
ya habíamos predicho. Vamos ahora a hacerlo con la composición del congreso \
que surgió de las elecciones de diciembre de 2015 (PP 123, PSOE 90, Podemos 6\
9, Cs 40, ERC 9, DL 9, PNV 6, IU 2, Bildu 2, CC 1) obtendremos las siguientes \
cuotas de poder: PP 37,73%, PSOE 20,98%, Podemos 20,98%, C's 8,37%. Todos los \
demás partidos tienen una cuota de poder inferior al 5%. Es interesante desta\
car que PSOE y Podemos tienen la misma cuota de poder, pero tal vez más inter\
esante es ver la cuota de poder obtenida por el pacto PSOE-C's. Si los conside\
ramos como una única coalición, es decir eliminamos al PSOE y C's y lo cambi\
amos por un único partido con 130 diputados resulta que las cuotas de poder d\
e PP, Podemos y la coalición PSOE-C's es identica, ya que es necesario y sufi\
ciente que se unan dos de ellos para formar gobierno y todos los otros partido\
s han reducido a cero su cuota de poder pues sin el apoyo de dos de los tres g\
rupos grandes (PP, Podemós, PSOE+C's) es imposible sumar 176 diputados. Si hi\
cieramos lo mismo con una coalición PSOE-Podemos-UP nos encontraríamos que l\
a cuota de poder de esta coalición sería de un 45,23%, mientras que la cuota\
de poder del PP y C's se habría igualado a un 15,71% y las de ERC, DiL y PNV\
estarían entorno al 8% y las de Bildu y CC en el 0,5%, porque podrían sumar\
para completar los 176 diputados. En cualquier caso, salvo abstenciones impro\
bables, todo lo que no sea sumar 176 diputados no hace un gobierno: Moraleja, \
tal vez para que haya amor basta con dos pero como diría alguién que no recu\
erdo bien: no se trata de amor sino de sexo, idiota. Lo que pasó fue que no s\
e consiguió formar gobierno y se repitieron las elecciones. En el futuro tend\
remos que acostumbrarnos a los ménage à trois, o incluso a orgías más mult\
itudinarias.\000\000"
```

El método que hemos visto es válido para cualquier alfabeto. Vamos a escribir aquí el algoritmo para encontrar la clave de un texto encriptado con el Criptosistema de César por bloques de longitud l, para un texto escrito con un alfabeto con q elementos.

Algoritmo 1. Búsqueda de una clave de César por Bloques

ENTRADA: cf, l, n.
(cf = Lista de números entre 0 y $q^l - 1$, l = longitud de los bloques, q = cardinal del alfabeto);
$CF = cf$; $c = 0$;
Para i entre 0 y $l - 1$.
$cfp := CF \mod q$;

Ahora vamos a descifrar un texto escrito con el alfabeto de 256 caracteres de GAP que se ha encriptado con el criptosistema de Vigenère. En el proceso de cifrado primero hemos convertido el texto y la clave en dos listas de elementos de $\mathbb{Z}_{256}$ y al encriptar no transferimos la lista de números en símbolos. Por tanto el texto cifrado es simplemente la siguiente lista de elementos de $\mathbb{Z}_{256}$: No vamos a revelar de momento ni el texto en claro ni la clave que hemos llamado `texto` y `clave` respectivamente. Simplementes ponemos aquí los programas que hemos construido para cifrar y descifrar con el criptosistema de Vigenère:

```
Vigenere := function(texto,clave)
local NTexto,NClave,LClave,NCifrado;
NTexto := List(texto,IntChar);
NClave := List(clave,IntChar);
LClave := Length(clave);
NCifrado := List([0..Length(texto)-1],
x -> (NTexto[x+1] + NClave[(x mod LClave)+1]) mod 256);
return NCifrado;
end;

DescifrarVigenere := function(cf,clave)
local LClave;
LClave := Length(clave);
return  List([0..Length(cf)-1],
x -> CharInt((cf[x+1] - clave[(x mod LClave)+1]) mod 256) );
end;
```

y el resultado que ha salido cuando hemos cifrado `texto` con la clave `clave`:

```
gap> cf:=Vigenere(texto,clave);
[ 141, 205, 98, 201, 178, 235, 213, 97, 207, 170, 194, 177, 218, 226,
181, 209, 98, 198, 182, 153, 233, 175, 140, 178, 211, 178, 224, 230,
162, 217, 163, 129, 167, 222, 148, 169, 225, 175, 208, 181, 153, 229,
182, 209, 98, 212, 168, 153, 217, 174, 213, 182, 202, 6, 44, 148, 166,
218, 98, 166, 182, 233, 213, 4, 29, 163, 129, 179, 232, 230, 97, 216,
163, 129, 166, 218, 216, 166, 218, 163, 129, 147, 218, 230, 162, 217,
177, 214, 177, 237, 148, 132, 219, 175, 198, 167, 242, 148, 177, 219,
180, 129, 166, 218, 214, 173, 209, 113, 212, 164, 237, 55, 234, 216,
```

171, 213, 168, 167, 148, 148, 209, 98, 196, 178, 230, 228, 176, 218, 5,
14, 164, 153, 216, 166, 140, 184, 194, 181, 226, 213, 180, 140, 181,
198, 166, 220, 221, 176, 218, 167, 212, 111, 153, 215, 176, 218, 98,
209, 172, 222, 238, 162, 223, 98, 196, 178, 235, 232, 162, 223, 98,
137, 182, 228, 217, 181, 207, 170, 198, 182, 162, 148, 186, 140, 163,
207, 172, 230, 213, 164, 213, 177, 207, 168, 236, 160, 97, 207, 163,
211, 164, 220, 232, 166, 222, 171, 219, 164, 221, 213, 180, 140, 178,
208, 181, 153, 217, 173, 140, 170, 214, 176, 232, 230, 97, 205, 164,
212, 184, 235, 216, 176, 140, 187, 129, 182, 238, 230, 179, 209, 163,
205, 172, 236, 232, 162, 154, 98, 165, 168, 153, 217, 174, 213, 181,
202, 6, 44, 226, 97, 217, 167, 207, 182, 238, 213, 173, 152, 98, 212,
168, 153, 217, 180, 224, 180, 198, 177, 218, 214, 162, 140, 178, 211,
178, 224, 230, 162, 217, 163, 129, 168, 229, 148, 177, 222, 171, 206,
168, 235, 148, 165, 219, 175, 202, 177, 224, 227, 97, 208, 167, 129,
166, 218, 216, 162, 140, 175, 198, 182, 153, 217, 175, 140, 146, 194,
181, 218, 225, 176, 225, 176, 213, 99, 188, 227, 174, 209, 166, 218,
113, 153, 185, 175, 140, 174, 194, 99, 218, 215, 181, 225, 163, 205,
172, 221, 213, 165, 140, 181, 198, 99, 222, 225, 170, 224, 167, 207,
99, 235, 217, 177, 209, 182, 202, 166, 226, 227, 175, 209, 181, 129,
188, 153, 231, 172, 209, 182, 196, 171, 222, 231, 97, 223, 183, 198,
175, 237, 227, 180, 140, 180, 198, 179, 218, 230, 181, 213, 166, 208,
182, 153, 228, 176, 222, 98, 205, 164, 153, 228, 179, 219, 169, 211,
164, 230, 213, 164, 213, 5, 20, 177, 153, 216, 166, 216, 98, 196, 164,
231, 213, 173, 154, 98, 166, 177, 153, 224, 162, 140, 165, 194, 167,
222, 226, 162, 140, 166, 198, 99, 237, 217, 173, 209, 184, 202, 182,
226, 55, 244, 218, 98, 173, 178, 220, 213, 173, 213, 163, 129, 183,
218, 225, 163, 213, 5, 10, 177, 153, 231, 166, 140, 167, 206, 172, 237,
221, 166, 222, 177, 207, 99, 235, 217, 177, 219, 181, 202, 166, 226,
227, 175, 209, 181, 129, 167, 222, 228, 166, 218, 166, 202, 168, 231,
216, 176, 140, 166, 198, 99, 229, 213, 97, 220, 180, 208, 170, 235,
213, 174, 205, 165, 202, 6, 44, 226, 97, 209, 176, 129, 166, 218, 216,
162, 140, 165, 208, 176, 238, 226, 170, 208, 163, 197, 99, 242, 148,
181, 205, 175, 195, 172, 60, 29, 175, 140, 181, 198, 99, 222, 225, 170,
224, 171, 36, 246, 153, 228, 176, 222, 98, 198, 175, 153, 215, 162,
218, 163, 205, 99, 198, 200, 151, 140, 135, 212, 179, 218, 55, 242,
205, 112, 129, 136, 236, 232, 4, 13, 98, 196, 181, 222, 213, 165, 219,
110, 129, 167, 226, 230, 170, 211, 171, 197, 178, 153, 237, 97, 220,
180, 208, 183, 218, 219, 176, 218, 171, 219, 164, 221, 227, 97, 220,
177, 211, 99, 222, 224, 97, 212, 183, 206, 178, 235, 221, 180, 224,
163, 129, 164, 229, 214, 162, 207, 167, 213, 168, 60, 37, 176, 140,
140, 208, 164, 234, 233, 4, 25, 176, 129, 149, 222, 237, 166, 223, 110,
129, 164, 242, 233, 165, 205, 166, 208, 99, 233, 230, 170, 218, 165,
202, 179, 218, 224, 174, 209, 176, 213, 168, 153, 228, 176, 222, 98,
166, 181, 231, 217, 180, 224, 177, 129, 150, 222, 234, 170, 216, 174,

```
194, 111, 153, 198, 162, 47, 252, 205, 99, 188, 221, 174, 205, 181,
141, 99, 188, 213, 179, 216, 177, 212, 99, 186, 230, 166, 207, 167,
212, 99, 242, 148, 139, 225, 174, 202, 6, 26, 226, 97, 184, 5, 20, 179,
222, 238, 111, 140, 133, 194, 167, 218, 148, 177, 222, 177, 200, 181,
218, 225, 162, 140, 167, 211, 164, 153, 228, 179, 209, 181, 198, 177,
237, 213, 165, 219, 98, 202, 176, 226, 232, 162, 218, 166, 208, 99,
218, 148, 182, 218, 98, 209, 168, 235, 231, 176, 218, 163, 203, 168,
153, 218, 162, 217, 177, 212, 178, 165, 148, 178, 225, 167, 129, 185,
218, 148, 165, 205, 176, 197, 178, 153, 228, 162, 223, 177, 129, 164,
153, 224, 162, 223, 98, 197, 172, 236, 232, 170, 218, 182, 194, 182,
153, 231, 166, 207, 165, 202, 178, 231, 217, 180, 154, 98, 166, 177,
153, 224, 162, 140, 181, 198, 166, 220, 221, 4, 31, 176, 129, 183, 222,
231, 181, 213, 175, 208, 177, 226, 227, 180, 140, 163, 209, 164, 235,
217, 164, 47, 239, 194, 99, 232, 232, 179, 219, 98, 209, 168, 235, 231,
176, 218, 163, 203, 168, 153, 218, 162, 217, 177, 212, 178, 167, 148,
134, 223, 182, 198, 99, 60, 46, 173, 224, 171, 206, 178, 153, 228, 179,
209, 181, 198, 177, 237, 213, 163, 205, 98, 198, 175, 153, 231, 170,
211, 183, 202, 168, 231, 232, 166, 140, 178, 211, 178, 224, 230, 162,
217, 163, 143, 99, 198, 233, 164, 212, 163, 212, 99, 221, 217, 97, 216,
163, 212, 99, 220, 227, 179, 224, 171, 207, 172, 229, 224, 162, 223,
98, 197, 168, 229, 148, 177, 222, 177, 200, 181, 218, 225, 162, 140,
168, 214, 168, 235, 227, 175, 140, 166, 202, 165, 238, 222, 162, 208,
163, 212, 99, 233, 227, 179, 140, 133, 194, 181, 229, 227, 180, 140,
131, 211, 168, 220, 217, 180, 152, 98, 213, 164, 230, 214, 170, 47,
235, 207, 99, 220, 227, 175, 219, 165, 202, 167, 232, 148, 177, 219,
180, 129, 182, 238, 148, 177, 205, 180, 213, 172, 220, 221, 177, 205,
165, 202, 6, 44, 226, 97, 209, 176, 129, 175, 218, 148, 179, 209, 184,
202, 182, 237, 213, 97, 177, 174, 129, 141, 238, 217, 183, 209, 181,
143, 99, 190, 226, 97, 158, 114, 145, 122, 153, 192, 162, 140, 170,
208, 181, 218, 148, 164, 212, 163, 207, 164, 231, 232, 166, 140, 167,
207, 183, 235, 55, 244, 140, 167, 207, 99, 238, 226, 97, 220, 167, 211,
172, 232, 216, 176, 140, 166, 198, 99, 226, 226, 164, 209, 180, 213,
172, 221, 233, 174, 206, 180, 198, 99, 222, 226, 97, 209, 174, 129,
180, 238, 217, 97, 218, 177, 129, 182, 222, 148, 179, 209, 163, 205,
172, 243, 213, 179, 219, 176, 129, 177, 238, 217, 183, 219, 181, 129,
168, 233, 221, 180, 219, 166, 202, 178, 236, 162, 97, 177, 176, 129,
182, 222, 228, 181, 213, 167, 206, 165, 235, 217, 97, 208, 167, 129,
168, 236, 217, 97, 205, 5, 18, 178, 165, 148, 166, 216, 98, 206, 172,
236, 225, 176, 140, 167, 210, 184, 226, 228, 176, 140, 166, 198, 99,
218, 215, 181, 219, 180, 198, 182, 153, 232, 179, 205, 181, 205, 164,
221, 55, 244, 140, 174, 194, 99, 223, 55, 244, 222, 175, 214, 175, 218,
148, 181, 209, 174, 198, 185, 226, 231, 170, 226, 163, 129, 164, 153,
192, 162, 140, 116, 129, 167, 222, 148, 149, 209, 174, 198, 185, 226,
231, 170, 47, 245, 207, 99, 190, 231, 177, 205, 5, 18, 178, 229, 213,
```

```
97, 207, 177, 207, 99, 231, 233, 166, 226, 177, 212, 99, 220, 213, 177,
47, 239, 213, 184, 229, 227, 180, 140, 164, 194, 173, 232, 148, 166,
216, 98, 207, 178, 230, 214, 179, 209, 98, 197, 168, 153, 193, 182,
207, 170, 194, 166, 225, 213, 165, 205, 98, 207, 184, 226, 162, 97,
188, 163, 211, 164, 230, 227, 182, 218, 182, 129, 134, 232, 225, 166,
208, 187, 141, 99, 233, 230, 176, 220, 171, 198, 183, 218, 230, 170,
205, 98, 197, 168, 153, 224, 176, 223, 98, 197, 168, 235, 217, 164,
212, 177, 212, 99, 221, 217, 97, 184, 163, 129, 171, 232, 230, 162,
140, 165, 201, 164, 231, 213, 175, 224, 167, 141, 99, 226, 225, 177,
213, 166, 202, 6, 44, 148, 178, 225, 167, 129, 182, 222, 148, 182, 223,
163, 211, 164, 153, 217, 173, 140, 175, 202, 182, 230, 227, 97, 218,
177, 206, 165, 235, 217, 109, 140, 177, 129, 177, 232, 225, 163, 222,
167, 212, 99, 221, 217, 97, 216, 163, 212, 99, 236, 217, 164, 207, 171,
208, 177, 222, 231, 111 ]
```

Si supiéramos la longitud de la clave con la que se ha cifrado el texto en claro podríamos hacer una ataque para análisis de frecuencias. Para conseguir la longitud de la clave vamos a usar la entropía. La idea es que un texto en claro tiene menos entropía que un texto cifrado con una clave de Vigenère de longitud mayor que 1. Por ejemplo, calculamos la entropía normalizada del texto en claro, que todavía mantenemos en secreto y el texto cifrado. Para calcularla hemos implmentado en GAP una función `EntropiaLista` que usa la función `Entropia` que fue implementada al final de la Sección 1.3.

```
Entropia := function(x)
local y;
y := Filtered(x,u->u>Float(0));
return -Sum(y,u->u*Log(u)/Log(Float(Length(x))));
end;

EntropiaLista := function(x)
local Simbolos,fr;
Simbolos := SSortedList(x);
fr := List(Simbolos,a->Float(Length(Positions(x,a))))/Length(x);
return Entropia(fr);
end;
```

Al aplicar la segunda función a una lista nos proporciona la entropía de la distribución de frecuencias con la que aparecen los diferentes miembros de la lista.

```
gap> EntropiaLista(texto);
0.77525
gap> EntropiaLista(cf);
0.892243
```

Observamos que efectivamente la entropía del texto cifrado es mayor.

Para descubrir la longitud de la clave vamos a usar que si la longitud de la clave de Vigenère es l entonces cada uno de las sublistas formados por las letras que están en la posiciones congruentes módulo l tienen que tener una entropía similar al texto en claro. Por tanto vamos a calcular la entropía media de las l sublistas para diferentes valores de l hasta obtener una entropía que sea menor que divisiones en otras sublistas. Para eso construimos una función que cree las sublistas a partir del mensaje cifrado.

```
Sublistas := function(lista,l)
local n,r,q,TrozoLista,trozos,x;

n := Length(lista);
r := n mod l;
q := (n-r)/l;
###########################
TrozoLista := function(x)
if x <= r then
return List([0..q],y->lista[x+l*y]);
else
return List([0..q-1],y->lista[x+l*y]);
fi;
end;
###########################
return List([1..l], x -> TrozoLista(x));
end;
```

Ahora calculamos las entropías medias de las sublistas:

```
gap> for l in [1..20] do
> trozos := Sublistas(cf,l);
> Print("\n",[l,Sum(List(trozos,EntropiaLista))/l]);
> od;

[ 1, 0.892243026295506 ]
[ 2, 0.9152951659667639 ]
[ 3, 0.9268447316180073 ]
[ 4, 0.9382880178075352 ]
[ 5, 0.9450986494401937 ]
[ 6, 0.9423728697029459 ]
[ 7, 0.847927271505534 ]
[ 8, 0.9519529603152597 ]
[ 9, 0.9515666878959893 ]
[ 10, 0.9565625251755625 ]
[ 11, 0.958167863748157 ]
[ 12, 0.957034281683333 ]
[ 13, 0.961768632557037 ]
```

```
[ 14, 0.8795308313819081 ]
[ 15, 0.9609135324098413 ]
[ 16, 0.9650701681995213 ]
[ 17, 0.9617019132918946 ]
[ 18, 0.9631199544428986 ]
[ 19, 0.9664224571910026 ]
[ 20, 0.96842968557701 ]
```

Analizando los valores obtenidos para las entropías medias observamos que para $l = 7$, la entropía desciende considerablemente con respecto a las entropías medias para valores de l cercanos. En consecuencia sospechamos que la longitud de la clave de Vigenère será 7. Tomamos por tanto las sublistas obtenidas con $l = 7$. Vamos a escribir sólo la primera:

```
gap> trozos := Sublistas(cf,7);;
gap> trozos[1];
[ 141, 97, 181, 175, 162, 169, 182, 174, 166, 4, 97, 166, 162, 132, 177,
173, 234, 148, 176, 166, 180, 176, 176, 162, 162, 181, 186, 164, 97,
166, 180, 173, 97, 176, 179, 162, 174, 97, 173, 180, 162, 162, 177,
165, 97, 162, 175, 176, 174, 175, 181, 165, 170, 177, 175, 172, 97,
180, 181, 176, 179, 164, 166, 173, 162, 162, 173, 244, 173, 163, 166,
166, 177, 175, 166, 176, 97, 174, 97, 162, 170, 181, 175, 170, 176,
162, 151, 242, 4, 165, 170, 97, 176, 97, 97, 180, 162, 176, 4, 166,
165, 170, 174, 176, 180, 170, 162, 174, 179, 166, 139, 97, 111, 177,
162, 179, 165, 162, 182, 176, 162, 178, 165, 162, 162, 170, 166, 180,
162, 4, 181, 180, 164, 179, 176, 162, 134, 173, 179, 163, 170, 166,
162, 164, 97, 179, 162, 177, 162, 175, 162, 179, 180, 180, 170, 175,
177, 177, 177, 97, 179, 97, 183, 97, 162, 164, 166, 244, 97, 176, 164,
174, 97, 97, 179, 179, 183, 180, 97, 181, 97, 97, 166, 176, 176, 181,
179, 244, 244, 181, 170, 162, 149, 170, 177, 97, 166, 177, 180, 166,
179, 182, 165, 97, 182, 166, 176, 170, 176, 164, 97, 162, 175, 177,
178, 182, 173, 97, 109, 163, 97, 164, 111 ]
```

Una vez que conocemos la longitud de la clave vamos intentar buscar cuál podría ser la clave usando que lo más normal es que el símbolo más repetido sea el espacio cuyo correspondiente numérico es 32.

```
gap> IntChar(' ');
32
```

Construimos una función que calcula el elemento más frecuente de una lista y le resta 32 con la esperanza que ese sea el correspondiente numérico de la clave de César usada para cifrar esa lista. Después le aplicamos esto a cada una de las sublistas para obtener los correspondientes numéricos de una posible clave de Vigenère que hemos llamado `ClavePosible`

```
MasFrecuente := function(x)
local simbolos,fa;
```

```
simbolos := SSortedList(x);
fa := List(simbolos,s -> Length(Positions(x,s)));
return simbolos[ Position(fa,Maximum(fa)) ] - 32;
end;

gap> ClavePosible := List(trozos,MasFrecuente);
[ 65, 108, 66, 97, 67, 121, 116 ]
```

Utilizamos pues `ClavePosible` para descifrar nuestro mensaje cifrado:

```
gap> DescifrarVigenere(cf,ClavePosible);
"La hora chanante es un programa de humor que se emitió en España por la\
cadena Paramount Comedy por cable/satélite. Se componía de varias secci\
ones, con piezas cortas (sketches) y animaciones, caracterizadas por el hu\
mor absurdo y surrealista. De emisión mensual, se estrenaba programa el p\
rimer domingo de cada mes en Paramount Comedy. En la actualidad se emiten \
repeticiones y sketches sueltos repartidos por la programación del canal.\
En la cadena de televisión Localia también se emitieron reposiciones de\
pendiendo de la programación en cada comunidad y también se emitió por \
el canal MTV España. Está creado, dirigido y protagonizado por el humori\
sta albaceteño Joaquín Reyes, ayudado principalmente por Ernesto Sevilla\
, Raúl Cimas, Carlos Areces y Julián López. Cada programa era presentad\
o imitando a un personaje famoso, que va dando paso a las distintas seccio\
nes. En la sección testimonios aparecía otro personaje famoso. Este últ\
imo presentaba el siguiente programa. Muchas de las cortinillas del progra\
ma fueron dibujadas por Carlos Areces, también conocido por su participac\
ión en la revista El Jueves. En 2007 La hora chanante entró en un period\
o de incertidumbre en el que no se realizaron nuevos episodios. En septiem\
bre de ese año, el mismo equipo de actores trasladó la fórmula televisi\
va a La 2 de Televisión Española con nuevos capítulos bajo el nombre de\
Muchachada nui. Paramount Comedy, propietaria de los derechos de La hora \
chanante, impidió que se usara el mismo nombre, o nombres de las seccione\
s."
```

Este texto había sido copiado de la página de wikipedia dedicada al programa de humor "La Hora Chanante". Si quisiéramos podríamos tener el correspondiente numérico de la clave original que es una broma sobre la ciudad de varios de los creadores de "La Hora Chanante":

```
gap> List(ClavePosible,CharInt);
"AlBaCyt"
```

1.5 Seguridad perfecta

Supongamos fijado un criptosistema con conjunto de claves K, y conjuntos M_k y C_k de mensajes básicos claros y cifrados respectivamente para la clave k. Consideremos las variables aleatorias $\mathcal{K}$, $\mathcal{M}$ y $\mathcal{C}$ que tienen por espacio de sucesos respectivos los conjuntos K, de

claves, $\cup_{k\in K} M_k$, de mensajes en claro, y $\cup_{k\in K} C_k$, de mensajes cifrados. Entonces, $P(K=k)$, $P(\mathcal{M}=m)$ y $P(\mathcal{C}=c)$ denotan las probabilidades de que la clave sea k, el mensaje en claro sea m y el mensaje cifrado sea c, respectivamente. Un criptosistema se dice que tiene *seguridad perfecta* si las variables aleatorias $\mathcal{M}$ y $\mathcal{C}$ son independientes. Es decir un criptosistema tiene seguridad perfecta si cualquiera de las siguientes condiciones equivalentes se verifica:

- $P(\mathcal{M}=m,\mathcal{C}=c) = P(\mathcal{M}=m)P(\mathcal{C}=c)$.
- $P(\mathcal{M}=m|\mathcal{C}=c) = P(\mathcal{M}=m)$.
- $P(\mathcal{C}=C|\mathcal{M}=m) = P(\mathcal{C}=c)$.

(Ver Problema 3.)

Proposición 1.8 *Si un criptosistema tiene seguridad perfecta, entonces tiene al menos tantas claves con probabilidad no nula como mensajes básicos con probabilidad no nula.*

Demostración. Consideremos un criptosistema con seguridad perfecta. Sean $c_k : M_k \to C_k$ y $d_k : C_k \to M_k$ sus funciones de cifrado y descifrado para la clave k y sean K_p, M_p y C_p los conjuntos de claves, mensajes básicos en claro y mensajes básicos cifrados con probabilidad no nula. Si fijamos $c \in C_p$ y $m \in M_p$ entonces

$$\begin{array}{rcl} \displaystyle\sum_{k\in K, c_k(m)=c} P(K=k,\mathcal{M}=m) & = & \displaystyle\sum_{k, c_k(m)=c} P(K=k|\mathcal{M}=m)P(\mathcal{M}=m) = \\ P(\mathcal{C}=c|\mathcal{M}=m)P(\mathcal{M}=m) & = & P(\mathcal{C}=c)P(\mathcal{M}=m) \neq 0. \end{array}$$

Por tanto, para todo $m \in M_p$ existe $k_m \in K$ tal que $c_{k_m}(m) = c$ y $P(K=k_m,\mathcal{M}=m) \neq 0$. Esto último implica que $k_m \in K_p$. Además la aplicación $m \mapsto k_m$ es inyectiva pues si $k_{m_1} = k_{m_2}$, entonces $m_1 = d_{k_{m_1}}(c) = d_{k_{m_2}}(c) = m_2$. Por tanto $|K_p| \geq |M_p|$. ∎

Criptosistema de Vernam o de Libreta de Uso Único

Este criptosistema no es más que una ligera variación del criptosistema de Vigenère. La única diferencia es que la longitud de las claves básicas no está acotada. Es decir, se elige un alfabeto finito que se identifica con $\mathbb{Z}_n$ y el conjunto de claves es $\mathbb{Z}_n^\infty$. Para una clave $k = k_1 \dots k_m \in \mathbb{Z}_n^m$, de longitud n, el conjunto de mensajes en claro y cifrados es $M_k = C_k = \mathbb{Z}_n^m$ y se cifra como en el sistema de Vigenére: Si $x = x_1 \dots x_m \in \mathbb{Z}_n^m$ y $k = k_1 \dots k_m$ entonces

$$c_k(m) = k + m = (m_1 + k_1) \dots (m_n + k_n) \quad \text{y} \quad d_k(c) = c - k = (c_1 - k_1) \dots (c_n - k_n).$$

Otra peculiaridad de este criptosistema es que todos los mensajes han de tener longitud 1, es decir han de ser básicos, y la clave sólo se puede utilizar una vez (de ahí el nombre).

Proposición 1.9 *Si todos los mensajes en claro y todas las claves de la misma longitud tienen la misma probabilidad entonces el criptosistema de Vernam tiene seguridad perfecta.*

Demostración. Obsérvese que

$$P(\mathcal{M} = x|\mathcal{C} = y, K = k) = \begin{cases} 0, & \text{si } m + k \neq c; \\ 1, & \text{si } m + k = c. \end{cases}$$

Si x es un mensaje en claro de longitud l, entonces, como todos los mensajes en claro y todas las claves de longitud l tienen la misma probabilidad tenemos

$$\begin{aligned} P(\mathcal{M} = x) &= P(K = y - x) = P(K = y - x)P(M = x|C = y, K = y - x) \\ &= \textstyle\sum_{k \in K} P(K = k)P(M = x|C = y, K = k) = P(\mathcal{M} = x|\mathcal{C} = y). \end{aligned}$$

∎

La debilidad del criptosistema de Vernam es más de tipo logístico que de tipo criptográfico pues es necesario encontrar un método para que el cifrador y el descifrador utilicen la misma clave. Si tenemos que mandar la clave al igual que el mensaje perdemos la seguridad de que la clave no sea interceptada. Una alternativa es tener un método de generación de claves. En cualquier caso siempre necesitaremos un algoritmo de generación de las claves que nunca será completamente aleatorio.

1.6 DES Y AES

En 1977 el National Bureau of Standards (NBS) estableció el protocolo criptográfico Data Encryption System (DES) que se convirtió en un estándar de cifrado. DES sufrió ataques desde el principio por considerar algunos expertos que la longitud de la clave (56 bits) es demasiado corta lo que permite un ataque for fuerza bruta. Los primeros en observar esto, incluso antes de que el DES fuera publicado como un estándar, fueron M.W. Hellman y W. Diffie, por un lado, y D. Kahn, por otro. Hace ya algunos años que DES no es considerado completamente seguro por haber mostrado su debilidad frente a ataques masivos, algunos de ellos combinando la capacidad de muchos ordenadores. Una versión algo más segura es el triple DES, o TDES, que es una sofisticación del DES. Para suplir los defectos del DES en 2001 se estableció otro protocolo estándar conocido como AES (Advanced Encryption Standard). DES y AES usan el concepto de esquema de Lucifer y eso es lo que vamos a explicar en esta sección sin entrar en los detalles técnicos concretos de los diferentes protocolos[2] .

Los *esquemas de Lucifer* son criptosistemas basados en la dificultad de resolver ecuaciones polinómicas. Consideremos una aplicación polinómica

$$f : \mathbb{Z}_2^v \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$$

Obsérvese que en el polinomio f podemos suponer que los exponentes de cada indeterminada son todos ≤ 1, ya que $X^2 = X$ en $\mathbb{Z}_2$.

[2]Para ver los detalles sobre AES se pueden visitar la web del NIST (National Institute of Standards and Technology): https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf; https://csrc.nist.gov/publications/fips.

El conjunto de claves será $K = \mathbb{Z}_2^{(u-1)v}$ y los conjuntos de mensajes en claro y cifrados son $M = C = \mathbb{Z}_2^{2n}$. Cuando escribamos $m = m_1 \dots m_r \in \mathbb{Z}_2^{rs}$, estamos interpretando que hemos descompuesto m en r subpalabras de longitud s. Consideremos la aplicación de cifrado

$$\begin{array}{rcl} c : \mathbb{Z}_2^{(u-1)v} \times \mathbb{Z}_2^{2n} & \to & \mathbb{Z}_2^{2n} \\ (k = k_1 \dots k_{u-1}, m = m_0 m_1) & \mapsto & m_{u-1} m_u \end{array}$$

donde

$$m_{i+1} = m_{i-1} + f(k_i, m_i), \quad (1 \le i \le u).$$

Análogamente definimos la aplicación de descifrado

$$\begin{array}{rcl} d : \mathbb{Z}_2^{(u-1)v} \times \mathbb{Z}_2^{2n} & \to & \mathbb{Z}_2^{2n} \\ (k = k_1 \dots k_{u-1}, c = c_1 c_0) & \mapsto & c_u c_{u-1} \end{array}$$

donde

$$c_{i+1} = c_{i-1} + f(k_{u-i}, c_i), \quad (1 \le i \le d).$$

Ejemplo 1.10 Pongamos $n = 3$, $u = 5$, y $v = 3$, de forma que cada mensaje en claro o cifrado tiene la forma

$$m = m_1 m_2 \quad \text{con} \quad m_i = m_{i1} m_{i2} m_{i3} \quad (i = 1, 2).$$

y cada clave tiene la forma

$$k = k_1 k_2 k_3 k_4 \quad \text{con} \quad k_i = k_{i1} k_{i2} k_{i3} \quad (i = 1, 2, 3, 4).$$

Elegimos $f = (f_1, f_2, f_3) : \mathbb{Z}_2^2 \times \mathbb{Z}_2^3$ con

$$\begin{array}{rcl} f_1(k_1, k_2, k_3, x_1, x_2, x_3) & = & k_1(x_1x_3 + x_2) + k_2(x_1x_2 + x_1x_3) + k_3x_1x_3 + \\ & & k_1k_2(x_1x_2x_3 + x_3), \\ f_2(k_1, k_2, k_3, x_1, x_2, x_3) & = & k_1x_2 + k_2(x_1x_3 + x_2x_3) + k_1k_2(x_1x_2 + x_3) + k_1k_3(x_1 + x_2x_3), \\ f_3(k_1, k_2, k_3, x_1, x_2, x_3) & = & k_1(x_1 + x_2 + x_3) + k_2(x_2 + x_1x_3) + k_1k_2(x_2x_3 + x_3) + \\ & & k_2k_3(x_2 + x_1x_2x_3). \end{array}$$

Si fijamos la clave dada por

$$k_1 = 010, \quad k_2 = 101, \quad k_3 = 111, \quad k_4 = 110$$

entonces el mensaje $m = m_0 m_1 = 111110$ se cifra en cuatro pasos:

$$\begin{array}{rcl} m_0 & = & (1, 1, 1), \\ m_1 & = & (1, 1, 0), \\ m_2 & = & m_0 + f(k_1, m_1) = (1, 1, 1) + (f_1(0, 1, 1, 1, 1, 0), f_2(0, 1, 1, 1, 1, 0), f_2(0, 1, 1, 1, 1, 0)) \\ & = & (1, 1, 1) + (1, 0, 0) = (0, 1, 1), \\ m_3 & = & m_1 + f(k_2, m_2) = (1, 1, 0) + (f_1(1, 0, 1, 0, 1, 1), f_2(1, 0, 1, 0, 1, 1), f_2(1, 0, 1, 0, 1, 1)) \\ & = & (1, 1, 0) + (1, 0, 0) = (0, 1, 0) \\ m_4 & = & m_2 + f(k_3, m_3) = (0, 1, 1) + (f_1(1, 1, 1, 0, 1, 0), f_2(1, 1, 1, 0, 1, 0), f_2(1, 1, 1, 0, 1, 0)) \\ & = & (0, 1, 1) + (1, 1, 1) = (1, 0, 0) \\ m_5 & = & m_3 + f(k_4, m_4) = (0, 1, 0) + (f_1(1, 1, 0, 1, 0, 0), f_2(1, 1, 0, 1, 0, 0), f_2(1, 1, 0, 1, 0, 0)) \\ & = & (0, 1, 0) + (0, 0, 0) = (0, 1, 0) \end{array}$$

con lo que el mensaje cifrado es $c = m_4 m_5 = 100010$.

Veamos que efectivamente esto forma un criptosistema, es decir que

$$d_k(c_k(m_0m_1)) = m_0m_1.$$

Pongamos

$$c_1c_0 = c_k(m_0m_1) = m_{u-1}m_u$$

y mantengamos la notación de las definiciones de c y d. Vemos por inducción sobre i que $c_i = m_{u-i}$. Para $i = 0$ e $i = 1$, la igualdad se verifica por definición. Entonces aplicando la hipótesis de inducción para $i \geq 2$ tenemos

$$\begin{aligned} c_{i+1} &= c_{i-1} + f(k_{u-i}, c_i) = m_{u-(i-1)} + f(k_{u-i}, m_{u-i}) \\ &= m_{u-i+1} + f(k_{u-i}, m_{u-i}) = m_{u-i-1} + 2f(k_{u-i}, m_{u-i}) \\ &= m_{u-(i+1)} \end{aligned}$$

¿En que se basa la seguridad de los esquemas de Lucifer? Supongamos que conocemos parejas de mensajes en claro y mensajes cifrados (m_i, c_i). En última instancia $c_i = P(k_1, \ldots, k_{d-1}, m_i)$, donde P es algún polinomio. Para poder romper el sistema necesitamos encontrar ceros comunes de una familia de polinomios

$$c_i - P(X_1, \ldots, X_{d-1}, m_i)$$

y éste se convierte un problema difícil sin necesidad de que los grados de los polinomios sean grandes. De hecho tenemos el siguiente resultado de A.S. Fraenkel y Y. Yesha (1977) que podemos encontrar en el libro de M. Garey y D. Johnson[3].

Teorema 1.11 *El problema de decidir si un conjunto de polinomios con coeficientes en $\mathbb{Z}_2$ tienen una raíz común es $\mathbb{NP}$-completo.*

De momento este teorema no tiene sentido pues no se ha introducido el concepto de problema $\mathbb{NP}$-completo lo que se hará en la Sección 1.12.

1.7 Tiempo de Cálculo

En esta sección vamos a analizar una forma de medir el tiempo que se tarda en realizar un cálculo. En realidad esto depende de quién realize el cálculo, una persona, un ordenador con mayores o menores prestaciones, etc. Sin embargo, queremos que nuestra medida sea independiente de este tipo de consideraciones. El tiempo de cálculo también depende del método que se utilize para obtener el resultado, es decir del algoritmo que se utiliza para realizar el cálculo, con lo que en realidad debemos hablar de tiempo de cálculo de ejecución de un algoritmo. Finalmente nos interesa más tener una medida del comportamiento asintótico del tiempo de cálculo, es decir, queremos comparar cómo se va a comportar el tiempo de cálculo cuando la longitud de la entrada se hace grande.

Vamos a comenzar con una notación de comparación del comportamiento asintótico de las funciones.

[3]M. Garey y D. Johnson, Computers and intractability: a guide to the theory of $\mathbb{NP}$-completness, Freeman, San Francisco, 1979

Notación 1.12 *Dadas dos funciones $f, g : \mathbb{N} \to \mathbb{R}^+$, vamos a escribir:*

- $f = O(g)$*, si existe un* $C \in \mathbb{R}^+$ *tal que* $f(n) \leq Cg(n)$ *para todo* $n \in \mathbb{N}$*, o sea* $\frac{f(n)}{g(n)}$ *está acotada;*
- $f \sim g$*, si* $f = O(g)$ *y* $g = O(f)$*;*
- $f = o(g)$ *si* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Notaciones similares se pueden introducir para funciones de variable real pero la mayoría de las funciones que nosotros nos encontraremos estarán definidas en los números naturales.

La siguiente proposición es evidente. En ella hemos abusado de la notación y, en algún momento, hemos utilizado $f(n)$ para referirnos a una función f, entendiendo siempre que la variable independiente es n:

Proposición 1.13 *(1) La relación* $fRg \Leftrightarrow f = O(g)$ *es reflexiva y transitiva pero no es simétrica ni antisimétrica.*

(2) La relación $\sim$ *es una relación de equivalencia.*

(3) Si existe el límite $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ *entonces* $f = O(g)$

(4) Si $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}^+$*, entonces* $f \sim g$.

(5) Si f *es una función polinómica de grado* $\leq k$*, entonces* $f = O(n^k)$.

(6) $\log_a(n) \sim \log_b(n)$*,* $\log_a(n) = O(n^b)$ *y* $\log_a(n) \neq n^b = O(c^n)$ *y* $c^n \neq O(n^b)$*, para todo* $a, b, c > 0$*. Además,* $a \leq b$ *si y sólo si* $n^a = O(n^b)$ *si y sólo si* $a^n = O(b^n)$.

(7) Si $f = o(g)$ *y* $g = O(h)$*, entonces* $f + g = O(h)$.

El significado de la propiedad (6) es que existe una jerarquía de crecimiento entre las siguientes funciones de forma que las que están más a la derecha crecen más rápido que las que están más a la izquierda

$$\log_a(n) \quad n^{b_1} \quad n^{b_2} \quad c_1^n \quad c_2^n, \quad (a, b_1, b_2, c_1, c_2 \in \mathbb{R}^+ \text{ con } b_1 < b_2, c_1 < c_2).$$

A menudo en una expresión $f = O(g)$, las funciones f y g tienen más de una variable independiente. En tal caso es necesario precisar de qué variable estamos hablando. Veamos un ejemplo de esto.

Ejemplo 1.14 Consideremos las expresiones:

$$\sum_{i=1}^{n} i^k \qquad \text{y} \qquad \frac{n^{k+1}}{k+1}.$$

Si

$$f_1(n) = \sum_{i=1}^{n} i^k \qquad \text{y} \qquad g_1(n) = \frac{n^{k+1}}{k+1}$$

entonces considerando la integral de Riemann

$$\int_0^1 x^k dx$$

obtenemos que la n-ésima suma Riemanniana es

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{i}{n}\right)^k = \frac{f_1(n)}{(k+1)g_1(n)}$$

y por tanto

$$\lim_{n\to\infty}\frac{f_1(n)}{g_1(n)} = (k+1)\int_0^1 x^k dx = 1$$

con lo que $f_1 \sim g_1$.

Sin embargo, si ponemos

$$f_2(k) = \sum_{i=1}^{n} i^k \qquad \text{y} \qquad g_2(k) = \frac{n^{k+1}}{k+1}$$

entonces

$$\begin{array}{rcl} \lim_{k\to\infty}\frac{f_2(k)}{(k+1)g_2(k)} & = & \frac{1}{n}\lim_{k\to\infty}\sum_{i=1}^{n}\left(\frac{i}{n}\right)^k \\ & = & \frac{1}{n}\sum_{i=1}^{n}\lim_{k\to\infty}\left(\frac{i}{n}\right)^k = \frac{1}{n} \end{array}$$

y, por tanto

$$\lim_{k\to\infty}\frac{f_2(k)}{g_2(k)} = \infty,$$

de donde se deduce que $f_2 \neq O(g_2)$.

Ejemplo 1.15 Dados dos enteros positivos $b > 0$ y n denotaremos con $l_b(n)$ a la longitud de la expresión del número n en base b, es decir, al número de dígitos de la expresión de n en base b. Obsérvese que

$$l_b(n) = k \Leftrightarrow b^{k-1} \leq n < b^k,$$

con lo que

$$l_b(n) - 1 \leq \log_b(n) < l_b(n),$$

es decir

$$l_b(n) = \lfloor \log_b(n) \rfloor + 1,$$

donde $\lfloor x \rfloor$ denota el mayor entero menor o igual que el número real x. Por tanto

$$l_b(n) \sim \log_b(n) \sim \ln(n).$$

Ejemplo 1.16 (Tiempo de Cálculo de la Suma) ¿Cuál es el tiempo que tarda un ordenador en sumar 109 y 57? Ciertamente, esto depende de varios factores: velocidad del reloj de la máquina, tiempo que tarde en almacenar datos, etc. Si nos olvidamos de estos aspectos que están fuera de nuestro control, vamos a analizar el tiempo empleado en realizar un

cálculo o algoritmo, en función de una operaciones básicas que llamaremos operaciones bit. Las expresiones de los números 109 y 57 en base 2 son

$$109 \equiv 1101101, \quad 57 \equiv 111001.$$

La siguiente figura representa la suma de estos dos números.

$$\begin{array}{cccccccc} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & & {\scriptstyle 1} & \\ & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ + & & 1 & 1 & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}$$

Obsérvese que en cada paso del cálculo anterior se realiza una de las siguientes cuatro operaciones elementales, que llamaremos *operaciones bit*:

$$\begin{array}{rcl} 0+0+0 & = & 0 \\ 1+0+0 & = & 1 \\ 1+1+0 & = & 10 \\ 1+1+1 & = & 11 \end{array}$$

Si consideramos el tiempo necesario para hacer una operación bit como una unidad de tiempo, el tiempo que hemos necesitado para sumar 1101101 y 111001 es de 7 operaciones bit.

La *longitud* o el *número de bits* de un número es el número de dígitos de su expresión en base 2. El tiempo necesario para sumar dos números de longitudes k y l es $\max(k,l)$, en otras palabras:

$$\begin{array}{rcl} \text{Tiempo}[k-bit+l-bit] & = & \max(k,l) \\ \text{Tiempo}[m+n] & = & O(\max(\log m, \log n)) \end{array} \tag{1.3}$$

Ejemplo 1.17 (Tiempo de Cálculo de la Resta) Vamos ahora a calcular la diferencia de dos números:

$$\begin{array}{cccccccc} & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ - & & 1 & 1 & 1 & 0 & 0 & 1 \\ & {\scriptstyle 1} & {\scriptstyle 1} & & & & & \\ \hline & & 1 & 1 & 0 & 1 & 0 & 0 \end{array}$$

La tercera fila de la figura anterior toma cuenta de las "llevadas". Las operaciones elementales necesarias para realizar una resta, que añadiremos a la lista de las *operaciones bit*, son las siguientes:

$$\begin{array}{rcrcl} 0-(0+0) & = & 1-(1+0) & = & 0 \\ 1-(0+0) & & & = & 1 \\ 0-(1+0) & = & 1-(1+1) & = & 1-10 \\ 0-(1+1) & & & = & 0-10 \end{array}$$

El resultado es de la forma x ó $x-10$, donde x es el dígito que hay que colocar y la aparición de -10 significa que "nos llevamos 1".

Ejemplo 1.18 (Tiempo de Cálculo del Producto) Veamos ahora el tiempo necesario en el cálculo de un producto. Una forma de realizar un producto es el método escolar representado por el siguiente ejemplo, en el que suprimimos las "llevadas" para simplificar:

						1	1	0	1	1	0	1
			×				1	1	1	0	0	1
						1	1	0	1	1	0	1
			1	1	0	1	1	0	1			
			1	1	1	1	0	1	0	1	0	1
		1	1	0	1	1	0	1				
	1	0	1	0	1	0	1	0	0	1	0	1
	1	1	0	1	1	0	1					
1	1	0	0	0	0	1	0	0	0	1	0	1

Obsérvese que, si queremos multiplicar dos números de k y l bits (pongamos $l \leq k$), y despreciamos el tiempo empleado en mover a la izquierda el primer número, con este algoritmo necesitamos realizar un máximo de $l-1$ sumas de números de longitud $\leq l+k$. Por tanto

$$\begin{array}{lcl} \text{Tiempo}[k-bit \times l-bit] & = & O(l(k+l)) = O(lk) = O(k^2)(\text{ con } k \geq l) \\ \text{Tiempo}[n \times m] & = & O(\log^2 n)(\text{ con } n \geq m) \end{array} \tag{1.4}$$

Veamos un segundo algoritmo para realizar el producto de n y m, ambos de longitud menor o igual que k. Dividimos los dos números en dos de longitud $\leq \lceil \frac{k}{2} \rceil$. Para simplificar supondremos que k es par. Entonces lo que estamos haciendo es escribir

$$\begin{array}{rcl} n & = & n_1 2^{k/2} + n_2 \\ m & = & m_1 2^{k/2} + m_2. \end{array}$$

Entonces

$$\begin{array}{rcl} nm & = & n_1 m_1 2^k + (n_1 m_2 + n_2 m_1) 2^{k/2} + n_2 m_2 \\ & = & n_1 m_1 2^k + [n_1 m_1 + n_2 m_2 + (n_1 - n_2)(m_1 - m_2)] 2^{k/2} + n_2 m_2. \end{array}$$

Es decir reducimos el problema de realizar un producto de dos números de longitud $\leq k$ al de realizar tres productos de números de longitud $\leq \lceil k/2 \rceil$ y varias sumas de números de longitud menor o igual que k. Por tanto, si $T(k)$ es una estimación del tiempo necesario para multiplicar dos números de longitud $\leq k$, entonces

$$\begin{array}{l} T(k) \leq 3T(\lceil k/2 \rceil) + O(k) \leq 3^2 T(\lceil k/4 \rceil) + O(k) \leq \ldots \\ \leq 3^{\log_2 k} + O(k) = k^{\log_2 3} + O(k) = O(k^{\log_2 3}) \end{array}$$

Como $\log_2 3 \approx 1'59 < 2$, este algoritmo es un poco más rápido que el anterior.

Finalizamos esta sección con el tiempo de cálculo de la división entera que viene dado por la siguiente fórmula (ver el Problema 2):

$$\text{Tiempo}[k-bits : l-bits] = O((k-l)l) \quad (\text{con } k \geq l). \tag{1.5}$$

1.8 Algoritmo de Euclides

Como hemos visto en el tiempo necesario para calcular un producto depende del algoritmo empleado.

Un ejemplo más espectacular aparece cuando queremos calcular el máximo común divisor de dos números enteros. Si conocemos la factorización en producto de primos de los números en cuestión, el calculo es muy rápido. Sin embargo, si tenemos que calcular dicha factorización, el tiempo necesario se dispara. De hecho uno de las cosas que aprenderemos en este curso es que la búsqueda de un divisor de un número es una tarea lenta. Sin embargo, el algoritmo de Euclides nos proporciona un método rápido del cálculo del máximo común divisor.

Sean $a \geq b$ dos números enteros. El siguiente algoritmo proporciona como salida el máximo común divisor de dos enteros positivos a y b

Algoritmo 2. Algoritmo de Euclides

ENTRADA: a y b, enteros positivos.
$r_{-1} := \max(a, b)$, $r_0 := \min(b)$, $i = 0$.
Mientras que $r_i \neq 0$,
$i := i + 1$,
$r_i :=$ Resto de dividir r_{i-2}, entre r_{i-1}.
SALIDA: r_{i-1}.

Si a es un número real entonces las partes enteras por abajo y arriba de a se denotan $\lfloor a \rfloor$ y $\lceil a \rceil$ respectivamente. Es decir

$$\begin{aligned} \lfloor a \rfloor &= \text{Mayor entero menor o igual que } a, \\ \lceil a \rceil &= \text{Menor entero mayor o igual que } a. \end{aligned}$$

Así si a y b son enteros positivos entonces el cociente q y el resto r de dividir a entre b son

$$q = \left\lfloor \frac{a}{b} \right\rfloor, \quad r = a - bq.$$

Obsérvese que en la fórmula (1.5), $k - l$ coincide (salvo a lo máximo en una unidad) con la longitud del cociente. Por tanto, si denotamos por q_i el cociente de dividir r_{i-2} entre r_{i-1}, en i-ésimo paso del Algoritmo de Euclides, tenemos

$$q_i = \left\lfloor \frac{r_{i-2}}{r_{i-1}} \right\rfloor = \frac{r_{i-2} - r_i}{r_{i-1}},$$

y, por tanto, el tiempo necesario para realizar la i-ésima división es

$$O(\log q_i \log r_{i-1}) \leq O(\log(q_i) \log(b))$$

con lo que el tiempo total de cálculo es

$$O\left(\left(\sum_i \log q_i\right) \log b\right) = O\left(\log\left(\prod_i q_i\right) \log b\right).$$

Pero,

$$\prod_i q_i = \frac{a-r_1}{r_0}\frac{r_0-r_2}{r_1}\cdots = (a-r_1)\frac{r_0-r_2}{r_0}\frac{r_1-r_3}{r_1}\cdots \leq a \tag{1.6}$$

(obsérvese que $r_i \geq 1$, para todo resto menos para el último que puede ser 0, pero que no aparece en la cuenta anterior). Luego el tiempo total del Algoritmo de Euclides es

$$\begin{array}{lcll} \text{Tiempo}[\text{AE}(a,b)] & = & O(\log a \log b) = O(\log^2(a)), & (a \geq b) \\ \text{Tiempo}[\text{AE}(k-bits, l-bits)] & = & O(kl) = O(k^2)), & (k \geq l). \end{array} \tag{1.7}$$

En resumen, asintóticamente el tiempo de cálculo del máximo común divisor de dos números enteros utilizando el Algoritmo de Euclides es de un orden similar al del cálculo del producto de dichos números.

Ecuaciones diofánticas lineales

Con una ligera modificación, el Algoritmo de Euclides puede ser utilizado para resolver ecuaciones diofánticas del tipo

$$aX + bY = c. \tag{1.8}$$

Empezamos considerando el caso en que $c = 0$. Si ponemos $d = \text{mcd}(a, b)$, $a'd = a$, $b'd = b$, entonces la ecuación anterior equivale a

$$a'X_1 + b'Y_1 = 0.$$

y como $\text{mcd}(a', b') = 1$, entonces las soluciones de esta ecuación son las de la forma

$$X_1 = b'm, \qquad Y_1 = -a'm, \quad (m \in \mathbb{Z}).$$

Consideramos ahora el caso general, de la ecuación (1.8). Claramente, la ecuación tiene solución si y sólo si c pertenece al ideal (a, b) de $\mathbb{Z}$ generado por a y b que sabemos que es el ideal generado por d. Por tanto, lo primero que tenemos que hacer es decidir si d divide a c y si no es el caso deducir que la ecuación no tiene solución. Supongamos que d divide a c y supongamos y que conocemos dos enteros (x_0, y_0) que satisfacen

$$ax_0 + by_0 = d. \tag{1.9}$$

Entonces $X_0 = c'x_0$ e $Y_0 = c'y_0$ es una solución de la ecuación (1.8) y todas las soluciones serán de la forma

$$X = X_0 + X_1, \qquad Y = Y_0 + Y_1$$

con (X_1, Y_1) una solución de la ecuación $ax + by = 0$. Por tanto, la solución general de la ecuación 1.8 es

$$X = c'x_0 + b'm \qquad Y = c'y_0 + a'm, \quad (m \in \mathbb{Z}).$$

Sólo nos falta un método para calcular una solución (x_0, y_0) de la ecuación (1.9) y esto es lo que nos proporciona la siguiente modificación del Algoritmo de Euclides, que llamaremos *Algoritmo de Euclides Extendido*, en la que simplemente guardamos los cocientes de las divisiones efectuadas y acumulamos la información en las sucesiones (u_i) y (v_i).

Algoritmo 3. Algoritmo de Euclides Extendido

ENTRADA: a,b.
$r_{-1} := a$, $r_0 := b$, $u_{-1} = 1$, $v_{-1} = 0$, $u_0 = 0$, $v_0 = 1$, $i = 0$.
Mientras que $r_i \neq 0$,
$i := i + 1$,
$q_i := \left\lfloor \frac{r_{i-2}}{r_{i-1}} \right\rfloor$ (cociente de dividir r_{i-2} entre r_{i-1}),
$r_i := r_{i-2} - q_i r_{i-1}$ (resto de dividir r_{i-2}, entre r_{i-1}),
$u_i = u_{i-2} - q_i u_{i-1}$,
$v_i = v_{i-2} - q_i v_{i-1}$,
SALIDA: $d = r_{i-1}, x_0 = u_{i-1}, y_0 = v_{i-1}$.

Una versión más compacta es la siguiente:

Algoritmo 4. Algoritmo de Euclides Extendido

ENTRADA: a,b.
$A := a$; $B := b$; $u_0 := 1$; $v_0 := 0$; $u_1 := 0$; $v_1 := 1$;
Mientras que $B \neq 0$,
$q := \lfloor \frac{A}{B} \rfloor$; $r := A - qB$;
$A := B$; $B := r$;
$U := u_1$; $V := v_1$;
$u_1 = u_0 - qu_1$; $v_1 = v_0 - qv_1$,
$u_0 := U$; $v_0 := V$.
SALIDA: A, u_0, v_0.

Obsérvese que en cada paso q_i y r_i son el cociente y el resto de la división de r_{i-2} por r_{i-1}. Utilizando esto es fácil demostrar por inducción sobre i que

$$u_i a + v_i b = r_i,$$

con lo que, en el momento en el que el algoritmo se detiene, $d = r_{i-1} = \text{mcd}(a, b)$, $x_0 = u_{i-1}$, $y_0 = v_{i-1}$ y se verifica

$$x_0 a + y_0 b = d.$$

Tiempo de cálculo del Algoritmo de Euclides Extendido

Vamos a estimar el tiempo de cálculo del Algoritmo 3. Para ello empezaremos con el siguiente lema en el que estamos utilizando la notación del Algoritmo 3.

Lema 1.19 *Si* $i \geq 1$ *entonces* $r_{i+2} < \frac{r_i}{2}$ *y* $l(u_i) \leq \sum_{j=1}^{i} l(q_j) + i \leq l(a) + 1$.

Demostración. Si $r_{i+1} \leq \frac{r_i}{2}$, entonces $r_{i+2} < r_{i+1} \leq \frac{r_i}{2}$. En caso contrario $2r_{i+1} > r_i$ y, por tanto, el cociente de dividir r_i entre r_{i+1} es $q_{i+2} = 1$ y

$$r_{i+2} = r_i - r_{i+1} < r_i - \frac{r_i}{2} = \frac{r_i}{2}.$$

Esto demuestra la primera desigualdad del lema. Para demostrar la segunda razonamos por inducción sobre i. Claramente

$$l(u_1) = l(1) \leq l(q_1).$$

Aplicando la hipótesis de inducción tenemos para $i > 1$ que

$$\begin{array}{rcl} l(u_i) & = & l(u_{i-2} - q_i u_{i-1}) \\ & \leq & \max[l(u_{i-2}), l(q_i) + l(u_{i-1})] + 1 \\ & \leq & \max[\sum_{j=1}^{i-2} l(q_j) + i - 2, l(q_i) + \sum_{j=1}^{i-1} l(q_j) + i - 1] + 1 \\ & \leq & \sum_{j=1}^{i} l(q_j)] + i \end{array}$$

En la última desigualdad hemos utilizado (1.6). Finalmente, de (1.6) deducimos que $\sum_{j=1}^{i} l(q_j) + i \leq l\left(\prod_{j=1}^{i} q_j\right) + i \leq l(a) + i$. ∎

De la última desigualdad del Lema 1.19 deducimos que cada operación

$$u_i = u_{i-2} - q_i u_{i-1}$$

requiere un tiempo

$$O((l(a) + i)l(q_i))$$

y otro tanto sucede con cada operación

$$v_i = v_{i-2} - q_i v_{i-1}.$$

Como las otras dos operaciones se realizan en un tiempo de orden menor, cada bucle requiere un tiempo $O((l(a)+i)l(q_i))$. De la primera desigualdad del Lema 1.19 deducimos que el número total de bucles del algoritmo es $k \leq 2l(a)$, con lo que el tiempo de cálculo del algoritmo será

$$\begin{array}{rcl} O(\sum_{i=1}^{k}(l(a) + i)l(q_i)) & = & O(l(a)\sum_{i=1}^{k} l(q_i) + \sum_{i=1}^{k} il(q_i)) \\ & \leq & O\left(l(a)^2 + \left(\sum_{i=1}^{2l(a)} i\right)\left(\sum_{i=1}^{k} l(q_i)\right)\right) \\ & \leq & O\left(l(a)^2 + \frac{2l(a)(2l(a)+1)}{2} l(a)\right) = O(l(a)^3). \end{array}$$

Resumiendo

$$\text{Tiempo[Solución } ax + by = \text{mcd}(a, b)] = O(\log^3(a)) \quad (a \geq b)$$

ó

$$\text{Tiempo[Solución } ax + by = \text{mcd}(a, b)] = O(k^3) \quad l(a), l(b) \leq k.$$

Operaciones Modulares

En esta subsección n es un entero mayor que 1. Suponemos que cada elemento de $\mathbb{Z}_n$ viene representado por un número entre 0 y $n-1$. El tiempo de cálculo de la suma y el producto en $\mathbb{Z}_n$ viene dado por las siguientes fórmulas (ver el Problema 2):

$$\begin{array}{lcl} \text{Tiempo[Suma módulo } n] & = & O(\log^2 n) \\ \text{Tiempo[Producto módulo } n] & = & O(\log^2 n) \end{array}$$

Supongamos ahora que a y n son coprimos. Entonces a es inversible en $\mathbb{Z}_n$. Un inverso de a es un número x tal que existe una solución y de la ecuación

$$ax + ny = 1.$$

Por tanto una pequeña variación del algoritmo visto en la Sección 1.8 calculará el inverso de a módulo n y

$$\text{Tiempo[Inverso módulo } n] = O(\log^3 n).$$

1.9 Exponenciación doblando cuadrados

Vamos a ver ahora un método rápido para calcular potencias en un semigrupo. El método obvio para calcular a^n de multiplicar a por si mismo $n-1$ veces es demasiado lento ya que tenemos tantas operaciones como el valor de n con lo que es un algoritmo de tiempo exponencial. Vamos a presentar un método rápido para calcular potencias, denominado *exponenciación doblando cuadrados* y lo presentaremos en el caso general de que las cuentas se realizan en un semigrupo cualquiera. Dicho semigrupo podría ser el grupo aditivo de $\mathbb{Z}_n$, o su semigrupo multiplicativo, pero el método sirve para cualquier semigrupo.

Supongamos que tenemos un semigrupo S que denotaremos multiplicativamente y queremos calcular a^n con $a \in S$ y n un entero positivo. Comenzamos escribiendo el exponente n en base 2:

$$n = n_{l-1} \ldots n_2 n_1 n_0$$

es decir

$$n = n_0 + 2n_1 + 2^2 n_2 + \ldots + 2^{l-1} n_{l-1}, \text{ con cada } n_i \in \{0, 1\}.$$

Esta parte del cálculo consiste en hacer l divisiones entre 2, de números menores o iguales que b. Podemos suponer que este cálculo ya está hecho pues se entiende que los datos siempre vienen expresados en base 2. Entonces calculamos

$$r_t = (a^{2^t} \mod n), (t = 0, 1, \ldots, l-1).$$

Obsérvese que

$$r_t = (a^{2^t} \mod n) = ((b^{2^{t-1}})^2 mod n).$$

Por tanto tenemos que hacer l productos en S. Por tanto, el tiempo de cálculo será l veces lo que cueste hacer un producto en S. Finalmente hacemos

$$a^n = a^{n_0 + 2n_1 + 2^2 n_2 + \ldots + 2^{l-1} n_{l-1}} = a^{n_0} (a^2)^{n_1} (a^{2^2})^{n_2} \cdots (a^{2^{l-1}})^{n_{l-1}} \equiv r_0^{n_0} r_1^{n_1} r_2^{n_2} \cdots r_{l-1}^{n_{l-1}} \mod n.$$

Como cada n_i es 0 ó 1, este último cálculo consiste en el producto de l enteros módulo n. O sea tenemos que hacer un máximo de $l-1$ productos. En total hemos hecho unas $2l$ multiplicaciones en S. Una de las las ventajas del algoritmo anterior es que el cálculo de los r_t es aprovechable si necesitamos calcular varias potencias de a en S.

Algoritmo 5. Exponenciación doblando cuadrados

ENTRADA: a un elemento de un semigrupo, n un entero positivo.
$n_0 n_1 \dots n_l :=$ Expresión binaria de n;
$a_1 := a$;
$s := 1$;
Para i en $\{0, 1, \dots, l\}$
Si $n_i = 1$ entonces $s := sa_1$;
$a_1 := a_1^2$;
SALIDA: s.

El tiempo de cálculo del Algoritmo 5 depende del tiempo de cálculo de la multiplicación en el semigrupo. Si suponemos que el tiempo de cálculo de escribir n en base 2 es de orden menor que esta multiplicación sólo tendríamos que calcular el número de operaciones que tenemos que hacer en el semigrupo, que es del orden de la longitud del exponente. Por tanto,

$$\text{Tiempo}(-^{l-bit} \text{ en un semigrupo } S) = O(l)\text{Tiempo}(\text{Multiplicación en } S).$$

Por ejemplo, calcular a^n en $\mathbb{Z}_m$ nos lleva un tiempo

$$\text{Tiempo}[(a^n \mod n)] = O(\log n \log^2 n).$$

1.10 Cuerpos finitos

En esta sección vamos a discutir la estructura de los cuerpos finitos. La *característica* de un anillo A es el menor entero positivo n tal que $n1_A = 0$, si tal número existe. Si dicho número no existe se dice que A tiene característica 0. Obsérvese que si n es la característica de A, no sólo $n1_A = 0$ sino que también $na = 0$ para todo $a \in A$. Denotaremos la característica de A por Car A. Por ejemplo, Car $\mathbb{Z} = 0$ y, para todo entero positivo n, se tiene Car $\mathbb{Z}_n = n$.

Obsérvese que la aplicación $\mathbb{Z} \to A$, dada por $n \mapsto n1_A$ es un homomorfismo de anillos (de hecho es el único homomorfismo de anillos de $\mathbb{Z}$ a A), y su núcleo es el ideal de $\mathbb{Z}$ generado por la característica de A. Del Primer Teorema de Isomorfía se deduce que si A tiene característica n entonces A tiene un subanillo isomorfo a $\mathbb{Z}_n$. (Aquí estamos identificando $\mathbb{Z}_0$ con $\mathbb{Z}$.) De hecho éste es el menor subanillo de A que se llama *anillo primo* de A. En el caso particular en el que A sea un dominio, entonces $\mathbb{Z}_n$ es también un dominio, lo que implica que $n = 0$ o un número primo. Esto demuestra que la característica de un dominio de integridad es 0 o un número primo.

En particular, si F es un cuerpo finito, entonces la característica de F es necesariamente un número primo p. Si $p =$ Car F, entonces F contiene el cuerpo $\mathbb{Z}_p$ y F tiene una estructura

de espacio vectorial sobre $\mathbb{Z}_p$. Si n es la dimensión de este espacio vectorial, entonces F es isomorfo a $\mathbb{Z}_p^n$ como espacio vectorial y, por tanto, F tiene p^n elementos. Esto muestra que el cardinal de un cuerpo finito es una potencia de un número primo y este número primo es su característica.

Recuérdese que cada cuerpo K está contenido en un cuerpo algebraicamente cerrado F y que si $f \in K[X]$ es un polinomio entonces las raíces múltiples de f son los elementos de F que son raíces de f y de su derivada f'. Eso implica que si $f \neq 0$ entonces f tiene raíces múltiples en F si y sólo si f y f' no son coprimos.

Fijamos un cuerpo algebraicamente cerrado F de característica p y para cada $n \geq 0$ consideremos el siguiente conjunto

$$F_{p^n} = \{a \in F : a^{p^n} = a\}.$$

En primer lugar observemos que F_{p^n} es precisamente el conjunto de las raíces del polinomio $f = X^{p^n} - X$. Además este polinomio no tiene raíces múltiples. Eso es consecuencia de que $f = X(X^{p^n-1} - 1) = xg$ y de que la única raíz de $g' = (p^n - 1)X^{p^n-2}$ no es raíz de g. Por tanto, F_{p^n} tiene exactamente p^n elementos. Además F_{p^n} es un subcuerpo de F. En efecto, está claro que $0, 1 \in F_{p^n}$ y que si $a, b \in F_{p^n}$ entonces $-a, ab \in F_{p^n}$ y, si $a \neq 0$ entonces $a^{-1} \in F$. Por tanto sólo falta ver que $a + b \in F_{p^n}$. Para ello observamos que si $0 < i < p$ entonces p divide a $\binom{p}{i} = \frac{p!}{i!(p-i)!}$ lo que implica que

$$(a + b)^p = \sum_{i=0}^{p} \binom{p}{i} a^{p-i} b^i = a^p + b^p.$$

Razonando por inducción en n tenemos que

$$(a + b)^{p^n} = a^{p^n} + b^{p^n},$$

y por tanto, si $a, b \in F_{p^n}$, entonces $a + b \in F_{p^n}$. Esto prueba que F_{p^n} es un cuerpo con p^n elementos.

Hemos visto que el número de elementos de un cuerpo finito es una potencia de un primo y que para toda potencia de un primo p^n, hay al menos un cuerpo con p^n elementos. Más aún, dentro de cada cuerpo algebraicamente cerrado F de característica p hay un cuerpo con n elementos. En realidad, sólo hay uno. En efecto, si K es un subcuerpo de F con p^n elementos entonces K^* es un grupo de orden $p^n - 1$. Por tanto todos los elementos no nulos de K^* son raíces del polinomio $X^{p^n-1} - 1$ y todos los de K son raíces de $X^{p^n} - X = X(X^{p^n-1} - 1)$. O sea $K = F_{p^n}$. En otras palabras, el único subcuerpo de F con p^n elementos es exactamente el conjunto F_{p^n} formado por las raíces del polinomio $f = X^{p^n} - X$ en F. Además, está claro que éste es un cuerpo de descomposición sobre $\mathbb{Z}_p$ del polinomio f. Por un teorema de extensiones de cuerpos se tiene que todos los cuerpos de la forma F_{p^n}, con F un cuerpo algebraicamente cerrado, son isomorfos. De hecho, todos los cuerpos con p^n elementos son de esta forma pues estará contenido en algún cuerpo algebraicamente cerrado F y en consecuencia será precisamente F_{p^n}.

Obsérvese que las raíces de $X^{p^n} - X$ también son raíces de $X^{p^{nm}} - X$ para todo m. Por tanto, $F_{p^n} \subseteq F_{p^{nm}}$. Recíprocamente, si $F_{p^n} \subseteq F_{p^k}$, entonces F_{p^k} es un espacio vectorial de

dimensión finita sobre F_{p^n}, con lo que $p^k = |F_{p^k}|$ es una potencia de $p^n = |\mathbb{F}_{p^n}|$, o lo que es lo mismo k es múltiplo de n.

El siguiente teorema resume todo lo que hemos demostrado en los párrafos anteriores.

Teorema 1.20 *(1) Si F es un cuerpo finito entonces el cardinal de F es una potencia de un primo.*

(2) Para cada potencia de un primo q, existe un cuerpo con q elementos. Los elementos de dicho cuerpo son las raíces del polinomio $X^q - X$ en un cuerpo algebraicamente cerrado.

(3) Dos cuerpo finitos con el mismo cardinal son isomorfos.

(4) Si F es un cuerpo con q elementos y F' es un cuerpo con q' elementos, entonces F' contiene un subcuerpo isomorfo a F si y sólo si q' es potencia de q.

Nosotros supondremos que para cada primo p tenemos fijado una clausura algebraica $\widehat{\mathbb{Z}_p}$ de $\mathbb{Z}_p$ y para cada potencia q de p, denotaremos con $\mathbb{F}_q$ al cuerpo con q elementos contenido en $\widehat{\mathbb{Z}_p}$ que, por el Teorema 1.20, está formado por las raíces del polinomio $X^q - X$. Por ejemplo, $\mathbb{F}_p = \mathbb{Z}_p$. El mismo Teorema muestra que $\mathbb{F}_{p^n} \subseteq \mathbb{F}_{p^m}$ si y sólo si $n \mid m$.

Ahora nos queda la cuestión de cómo construir de forma explícita un cuerpo con $q = p^n$ elementos, para un primo p y un entero positivo n. Recuérdese que si E/F es una extensión de cuerpos y $\alpha \in E$ entonces $F(\alpha)$ denota el menor subcuerpo de E que contiene a F y a α. El homomorfismo de sustitución es la aplicación

$$\begin{array}{ccc} F[X] & \xrightarrow{S_\alpha} & E \\ f & \mapsto & f(\alpha) \end{array}$$

Si α es transcendente sobre F, entonces S_α es inyectivo y, por tanto, la imagen de S_α es isomorfa a $F[X]$. En tal caso $F(\alpha)$ es isomorfo al cuerpo de fracciones racionales $F(X)$. Sin embargo, el caso que más nos interesa es aquel en el que α es algebraico sobre F, o equivalentemente que S_α no es inyectivo. Como E es un cuerpo, el núcleo P de S_α es un ideal primo de $K[X]$. Si además α es algebraico, $P \neq 0$ y por tanto, P es un ideal maximal de $K[X]$ ya que $K[X]$ es un dominio euclídeo. En tal caso, la imagen de S_α es un cuerpo y $P = (\mathrm{Irr}(\alpha, F))$, donde $\mathrm{Irr}(\alpha, F)$ denota el (único) polinomio irreducible mónico de $F[X]$ que tiene a α como raíz, o lo que es lo mismo el polinomio mónico de grado mínimo que tiene a α como raíz. $\mathrm{Irr}(\alpha, F)$ se llama polinomio mínimo de α sobre F. Además, la dimensión de $F(\alpha)$ sobre F es el grado de $\mathrm{Irr}(\alpha, F)$.

Supongamos ahora que $q = p^n$, con n primo. Del Teorema del Elemento Primitivo (ver cualquier libro de Teoría de Galois) se tiene que $\mathbb{F}_q = \mathbb{F}_p(\alpha)$ para algún $\alpha \in \mathbb{F}_q$. Como $\mathbb{F}_q$ tiene dimensión n sobre $\mathbb{F}$, el polinomio $\mathrm{Irr}(\alpha, F_p)$ tiene grado n. Esto prueba lo siguiente.

Corolario 1.21 *Para todo $n \geq 1$, existe un polinomio irreducible f de grado n sobre $\mathbb{F}_p$. En tal caso $F[X]/(f)$ es un cuerpo con q elementos.*

Ejemplos 1.22 *Construcción de cuerpos finitos.*

(1) El polinomio X^2+X+1 es irreducible sobre $\mathbb{F}_2$, con lo que $\mathbb{F}_2[X]/(X^2+X+1)$ es un cuerpo con cuatro elementos, el único salvo isomorfismos. Identificaremos pues $\mathbb{F}_4$ con $\mathbb{F}_2[X]/(X^2+X+1)$. Si α denota la clase de X en $\mathbb{F}_4$, los cuatro elementos de $\mathbb{F}_2$ son $0, 1, \alpha$ y $1+\alpha$. Obsérvese que $\alpha^2 = 1+\alpha$ y $\alpha^3 = 1$, con lo que α es un elemento primitivo de $\mathbb{F}_4$.

(2) Para construir cuerpos con 8 y 9 elementos buscamos polinomios irreducibles sobre $\mathbb{F}_2$ y $\mathbb{F}_3$ de grados 3 y 2 respectivamente. Por ejemplo, X^3+X+1 es irreducible sobre $\mathbb{F}_2$ y X^2+1 es irreducible sobre $\mathbb{F}_3$, con lo que $\mathbb{F}_8 \simeq \mathbb{F}_2[X]/(X^3+X+1)$ y $\mathbb{F}_9 \simeq \mathbb{F}_3[X]/(X^2+1)$.

Obsérvese que si α representa la clase de X en $\mathbb{F}_{p^n} = \mathbb{F}_p[X]/f$, con f un polinomio irreducible de grado n, entonces $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ es una base de $\mathbb{F}_{p^n}$. En otras palabras, podemos identificar los elementos de $\mathbb{F}_{p^n}$ con polinomios de grado menor que n con coeficientes en $\mathbb{F}_p$. Ahora las operaciones aritméticas en $\mathbb{F}_{p^n}$ se reducen a operaciones con dichos polinomios que después serán reducidos módulo f. Utilizando esto no es difícil estimar el tiempo de cálculo de la suma, el producto y la división en cuerpos finitos. También se puede estimar el tiempo de cálculo de exponenciaciones en dichos cuerpos (ver Ejercicio 8).

El siguiente teorema puede encontrarse en la mayoría de los textos sobre cuerpos.

Teorema 1.23 *Un subgrupo finito del grupo de unidades de un cuerpo es cíclico.*

Un *elemento primitivo* de $\mathbb{F}_q$ es un generador de $\mathbb{F}_q^*$.

Ejemplos 1.24 *Elementos primitivos.*

(1) Un elemento primitivo de $\mathbb{F}_2$ es 1, de $\mathbb{F}_3$ es -1 y $\mathbb{F}_5$ tiene 2 elementos primitivos: 2 y 3.

(2) Si denotamos por α la clase de X en $\mathbb{F}_8 = \mathbb{F}_2[X]/(X^3+X+1)$ entonces α es un elemento primitivo de $\mathbb{F}_8$ pues las potencias de α son

$$\begin{array}{lll} \alpha = \alpha, & \alpha^2 = \alpha^2, & \alpha^3 = 1+\alpha, \\ \alpha^4 = \alpha+\alpha^2, & \alpha^5 = \alpha^2+\alpha^3 = 1+\alpha+\alpha^2 & \alpha^5 = \alpha+\alpha^2+\alpha^3 = 1+\alpha^2, \\ \alpha^6 = \alpha+\alpha^3 = 1. & & \end{array}$$

(3) Si α es la clase de X en $\mathbb{F}_{25} = \mathbb{F}_5[X]/(X^2-2)$, entonces $\alpha^2 = 2$ y $\alpha^4 = -1$, con lo que α tiene orden 8 y, por tanto, no es un elemento primitivo de $\mathbb{F}_{25}$. Ahora no es difícil ver que todos los elementos que no sean de la forma a o $a\alpha$ con $a \in \mathbb{F}_5^*$, (que son exactamente los que no están en el subgrupo generado por α), son elementos primitivos de $\mathbb{F}_{25}$.

1.11 Problemas y algoritmos

Problemas

Un *problema* es una descripción general de una tarea que depende de unos parámetros. Un *caso* o *especificación de un problema* es simplemente el problema obtenido al darle valor a los parámetros. Vamos a dar una definición más rigurosa.

Para empezar, para un conjunto no vacío A, que denominaremos alfabeto, vamos a denotar por A^∞ al conjunto de las sucesiones finitas de elementos de A, es decir

$$A^\infty = \cup_{n\in\mathbb{N}} A^n.$$

Los elementos de A^∞ los llamaremos *palabras* en el alfabeto A y los elementos de A^n son palabras de longitud n. El único elemento de A^0 es la palabra vacía $\emptyset$. Un *problema* en el alfabeto A es una aplicación $f : E \to B$, donde E es un subconjunto de A^∞. Los elementos de E son las especificaciones del problema. Para que quede descrito completamente el problema hay que decir si se trata de un problema de cálculo, de búsqueda o de decisión. En un *problema de cálculo* la tarea del problema consiste en calcular $f(e)$ para $e \in E$. En un *problema de búsqueda* $f : E \to B$, los elementos de B son conjuntos y dada una especificación e la tarea consiste en calcular un elemento de $f(e)$. Un problema de cálculo en el que el conjunto de llegada es $B = \{Verdadero, Falso\}$ se llama *problema de decisión*.

Veamos algunos ejemplos:

Ejemplos 1.25 *Problemas.*

(1) *Problema de Decisión de Primalidad.* Decidir si un número es primo o no. En este caso el alfabeto A puede estar formado por las cifras del sistema decimal, por 0 y 1, o por las cifras $0, 1, \ldots, b-1$ en la base b en la que deseemos representar los números naturales. Las especificaciones son los elementos de A^∞ que no comienzen por 0. Identificaremos cada especificación n con el número que representa y la aplicación que define el problema asocia a n, "Verdadero" si n es primo y "Falso" si n es compuesto.

(2) *Problema de Decisión de si un Número es Compuesto.* Tiene el mismo alfabeto y conjunto de especificaciones que el Problema de Decisión de Primalidad, pero la aplicación que lo define asocia los números primos con "Falso" y los compuestos con "Verdadero".

(3) *Problema de Búsqueda de un divisor propio de un número natural.* En este caso el alfabeto y conjunto de especificaciones del problema es el mismo que en los dos ejemplos anteriores y la aplicación que define el problema asocia n con el conjunto D_n de los divisores de n diferentes de 1 y n. Obsérvese que si n es primo entonces $D_n = \emptyset$ y, por tanto, el problema no tiene solución para esa especificación.

(4) *Problema de Factorización.* Éste es un problema de cálculo con las mismas especificaciones que los problemas anteriores, pero que asocia a cada entero positivo n una lista $p_1, \ldots, p_k$ de números primos tales que $n = p_1 \cdots p_k$, salvo que n sea 1 que es asociado con 1. Un caso particular que tendrá interés para nosotros será en el que el número es producto de dos primos distintos.

(5) *Problema del 3-coloreado.* Colorear un mapa con tres colores de forma que dos países consecutivos no tengan el mismo color. Este problema es un problema de búsqueda. El conjunto de especificaciones tiene que representar los países y las fronteras entre ellos. Es decir, una especificación tiene que definir la lista de las parejas de países que

tengan frontera común. Por ejemplo, podemos considerar como alfabeto las cifras en decimal, junto con tres símbolos mas: "(", ")" y ",". Una especificación es una sucesión del tipo $(n_1, m_1), \ldots, (n_k, m_k)$ donde cada n_i y m_i representan enteros positivos. Se entiende que cada n_i y m_i representa el número que le asociamos a un país. Denotemos con P el conjunto formado por todos los n_i y m_i de la especificación e, es decir el conjunto de los países. Entonces la aplicación que define el problema asocia e con el conjunto de las aplicaciones $\phi : I \to \{1, 2, 3\}$ que satisfacen que $\phi(n_i) \neq \phi(m_i)$ para todo $i = 1, \ldots, k$. Estas aplicaciones serán los 3-coloreados de la especificación. Por ejemplo, una especificación podría ser $(1, 2), (1, 3), (2, 4)$. En tal caso una posible solución del problema para esta especificación sería $1 \mapsto 1$, $2 \mapsto 2$, $3 \mapsto 2$ y $4 \mapsto 3$. Otra especificación podría ser $(1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4)$, que no tendría ninguna solución.

(6) *Problema de Decisión del 3-coloreado.* Es similar al caso anterior, pero lo que se trata de decidir es si el problema del 3-coloreado tiene solución o no para una especificación dada. Así la función que define el problema asociaría "Verdadero" a las especificación que tengan un 3-coloreado y "Falso" a las que no lo tengan.

(7) *Problema de los Circuitos Hamiltonianos.* Un grafo es un conjunto formado por puntos (vértices) y líneas que los unen, de forma que dos vértices pueden no estar unidos o estarlo por una o varias líneas. Un circuito Hamiltoniano es un camino continuo a través de todos los vértices del grafo que no pasa dos veces por la misma línea. El problema de los circuitos Hamiltonianos tiene dos versiones: el problema de decisión que consiste en decidir si un grafo admite un circuito Hamiltoniano y el de búsqueda que consiste en encontrar un circuito Hamiltoniano en caso de que exista.

(8) *Problema del Viajante.* Este es otro problema que tiene dos versiones, una de decisión y otra de cálculo. Los datos iniciales están formados por un conjunto de ciudades y las distancias entre ellas. En el problema de cálculo se trata de encontrar el camino más corto que pasa por todas las ciudades. En el problema de decisión se trata de decidir si hay un camino de longitud menor o igual que un número dado.

(9) *Problema de Decisión de Satisfabilidad.* Dada una proposición lógica en la que sólo se usen variables, los conectores de disyunción $\vee$, conjunción $\wedge$ y negación $\neg$ y paréntesis, decidir si existe una asignación de valores de verdad a las variables de forma que la proposición tiene valor verdadero. Por ejemplo, la proposición $(x \vee y) \wedge \neg x$ es satisfacible con la asignación de $x =$ "Falso" e $y =$ "Verdadero". Sin embargo la proposición $x \wedge (\neg x)$ no es satisfacible.

Algoritmos

Un *algoritmo* es una lista finita de instrucciones para resolver un problema. Las posibles entradas del algoritmo son las diferentes especificaciones del problema y el algoritmo tiene que tener una salida. Esta definición no es muy rigurosa pero vamos a admitirla por el momento.

Un algoritmo se dice que resuelve el problema de cálculo $P : E \to X$ si para cada especificación $e \in E$ la salida del algoritmo es $P(e)$. En el caso de un problema de búsqueda, la salida debe de ser no exactamente $P(a)$ sino un elemento de $P(a)$.

Un *certificado* de un problema de cálculo es un algoritmo que verifica que una solución del problema es correcta. Es decir, un certificado del problema $P : A \to X$ es un algoritmo que tiene como entrada una terna (a, x, d), con $a \in A$, $x \in X$ y d un dato adicional y resuelve en positivo el problema de decidir si se cumple $x = f(a)$, o sea si la salida del algoritmo es "Verdadero" entonces $x = f(a)$. En el caso de un problema de búsqueda lo que debe de verificar el certificado es que $x \in f(a)$. Un *certificado de un problema de decisión* $P : A \to \{Verdadero, Falso\}$ es un algoritmo que, con algún dato adicional, decide sobre la corrección de una solución positiva del problema. Es decir es un algoritmo en el que la entrada está formada por un elemento $a \in A$ y otro dato más y si la salida es "Sí", entonces $P(a) =$ Verdadero. Obsérvese que para un problema de decisión no es lo mismo un certificado, en el que el problema se está viendo como problema de cálculo, que si se está viendo como problema de decisión.

Ejemplos 1.26 *Certificados.*

(1) Consideremos el problema de decidir si un número es compuesto, es decir $P(n) = Verdadero$ si n es compuesto y $P(n) = Falso$, en caso contrario. Consideremos un algoritmo en el que la entrada son dos números naturales n y d y la salida es "SI" en el caso en el que d divida a n y $1 < d < n$ y "NO" en caso contrario. Si la salida de este algoritmo es positiva, entonces n es compuesto. Obsérvese que éste no es un certificado del problema de decisión sobre la primalidad de un número porque no sirve para asegurar si un número es primo.

(2) El algoritmo en el que la entrada está formada por tres números naturales n, p y q y la salida es SI, en el caso en el que $p \neq 1 \neq q$ y $n = pq$, es un certificado del Problema de Factorización de un número n que sea producto de dos primos distintos.

(3) En los problemas de búsqueda y decisión de un circuito Hamiltoniano, un certificado es un algoritmo que comprueba que un camino dado es efectivamente Hamiltoniano.

(4) En el Problema de Decisión del Viajante, un certificado es un algoritmo que comprueba que la distancia total de un camino dado es menor o igual que el número dado en la especificación del problema. En el Problema de Cálculo del Camino Más Corto un certificado requiere mucho más esfuerzo ya que requiere demostrar que todos los caminos que pasan por todas las sedes tienen longitud mayor o igual que la solución dada.

(5) Un certificado para el Problema de Decisión de Satisfabilidad, tiene como entrada la proposición y una asignación de verdad a cada una de las variables. El algoritmo verifica si con dicha asignación y aplicando las leyes lógicas la proposición toma el valor verdadero.

entonces asigna un valor de verdad a la proposición con la asignación de verdad natural: $\neg x$ tiene asignado V si y sólo si x tiene asignado F; $x \vee y$ tiene asignado V si y sólo si x ó y tienen asignado V y $x \wedge y$ tiene asignado V si y sólo si tiene valor V.

1.12 Tiempo polinomial

Un algoritmo se dice de *tiempo polinomial* si existe un $d \in \mathbb{N}$ tal que el tiempo de cálculo del algoritmo para una especificación de longitud k es $O(k^d)$. Obsérvese que si la especificación es un número n, escrito en una base arbitraria, entonces que el algoritmos tenga tiempo polinomial quiere decir que el tiempo de cálculo es $O(\log^d n)$ para algún $d \in \mathbb{N}$.

Un algoritmo se dice de *tiempo exponencial* si existe un número real $a > 1$ tal que el tiempo de cálculo del algoritmo para una especificación de longitud k es $O(a^k)$. A menudo se suele dar la definición diciendo que el tiempo de cálculo es $O(e^{ck})$ con $c > 0$. Si expresamos esto en términos de la entrada n, entonces el tiempo de cálculo es $O(a^{\log n})$ ó $O(e^{c \log n})$.

Hemos visto los siguientes ejemplos de algoritmos de tiempo polinomial:

- Sumas, productos, restas, divisiones en $\mathbb{Z}$.
- Cálculo de máximo común divisor y mínimo común múltiplo de números enteros.
- Resolución de ecuaciones diofánticas del tipo $ax + by = c$.
- Sumas, productos, restas, inversos, exponenciaciones en $\mathbb{Z}_n$.
- Cálculo en un cuerpo finito. Bueno, esto sólo lo habrá visto quien haya resuelto correctamente el Ejercicio 8.

La idea es que un algoritmo de tiempo polinomial es un algoritmo rápido y un algoritmo de tiempo no polinomial es un algoritmo lento. Sin embargo, a efectos prácticos, un algoritmo de tiempo de cálculo polinomial puede ser lento si la constante d es muy grande. Por ejemplo, para que un algoritmo de tiempo $O(n^{100})$ sea más rápido que uno de tiempo $O(e^{10^{-5}n})$ la entrada tiene que ser muy grande, lo que tal vez no se dé nunca en nuestros cálculos. Afortunadamente en todos los ejemplos que hemos visto el exponente calculado es bastante pequeño. Esto es lo que podríamos llamar de forma poco precisa tiempo polinomial efectivo.

El algoritmo que vimos para las exponenciaciones en $\mathbb{Z}_n$ tiene tiempo polinomial pero no lo tiene el algoritmo naïf que consiste en calcular la potencia y reducir módulo n. De hecho el algoritmo tiene tiempo de cálculo exponencial. Un algoritmo que tiene tiempo exponencial es el del cálculo de $n!$ de la forma obvia. (Calcular la constante c, para este problema).

Algunos algoritmos no son de tiempo polinomial pero su tiempo de cálculo es de orden estrictamente menor que una función exponencial. Para graduar los tiempos intermedios introducimos las siguientes funciones:

$$L_{\gamma;c}(k) = e^{c(\ln k)^{\gamma}(\ln \ln k)^{1-\gamma}}.$$

Un L_γ-*algoritmo* es un algoritmo cuyo tiempo de cálculo, para una especificación de longitud k, es $O(L_{\gamma;c}(k))$ para algún c. Obsérvese que un L_0-algoritmo es un algoritmo de tiempo polinomial y un L_1-algoritmo es un algoritmo de tiempo exponencial. Un L_γ-algoritmo, con $\gamma < 1$, se dice de *tiempo de cálculo subexponencial* .

Si un problema admite un algoritmo de tiempo polinomial que lo resuelve entonces decimos que el problema está en la *clase* $\mathbb{P}$. Un problema se dice que está en la *clase* $\mathbb{NP}$ si se puede ser resolver con un algoritmo sin límite de tiempo y admite un certificado de tiempo polinomial.

Todos los certificados de los Ejemplos 1.26 son de tiempo polinomial. Por tanto, los problemas de decisión de si un número es compuesto, de 3-coloreado, de decisión de si un grafo admite un camino hamiltoniano y de decisión de Satisfabilidad están en la clase $\mathbb{NP}$.

Obsérvese que en el caso en el que se trate de un problema de decisión el certificado ha de ser de solución positiva. Por ejemplo, el problema de decidir si un número es compuesto está en la clase $\mathbb{NP}$. Sin embargo no es tan fácil ver que el Problema de Decisión de Primalidad está en la clase $\mathbb{NP}$, aunque es cierto y veremos esto más adelante. Un ejemplo de un problema que está en la clase $\mathbb{NP}$, pero no se sabe si su negación está en la clase $\mathbb{NP}$ es el de los circuitos Hamiltonianos.

Está claro que

$$\mathbb{P} \subseteq \mathbb{NP}$$

ya que un algoritmo que resuelva un problema se puede transformar en un certificado de dicho problema. Sin embargo no se sabe si esta inclusión es estricta o si por el contrario las class $\mathbb{P}$ y $\mathbb{NP}$ son iguales. Es decir, la siguiente pregunta

$$¿\ \mathbb{P} = \mathbb{NP}\ ?$$

es un problema abierto y es de hecho unos de los siete problemas que fueron denominados como Problemas del Milenio en el año 2000 por Instituto Clay de Matemáticas. Para cada uno de estos problemas se estableció un premio de un millón de dólares.

Si P es un problema, un *oráculo* de P es una llamada a un algoritmo hipotético que resuelve P. Por supuesto, esta llamada al oráculo lleva incluido unos datos de entrada específicos para el algoritmo que resuelve P. Podemos pensar los oráculos de P como subrutinas dentro de un algoritmo que llaman a un algoritmo que resuelve P.

Sean P_1 y P_2 dos problemas. Decimos que P_2 se *reduce a* P_1 *en tiempo polinomial* si existe un algoritmo que resuelve P_2 utilizando varios oráculos de P_1, de forma que, descontando el tiempo de las llamadas a los oráculos de P_1, el tiempo de cálculo del algoritmo es polinomial. Dos *problemas* se dice que son *equivalentes en tiempo polinomial* si cada uno se reduce al otro en tiempo polinomial.

Un problema se dice que es $\mathbb{NP}$-*completo* si está en la clase $\mathbb{NP}$ y todos los problemas de la clase $\mathbb{NP}$ se reducen a él en tiempo polinomial. Un *problema* $\mathbb{NP}$-*completo* es un problema de decisión $\mathbb{NP}$-completo. Ejemplos de problemas $\mathbb{NP}$-completos son los problemas de Satisfabilidad, de Decisión del Viajante, Decisión de 3-Coloreado y de Existencia de Circuitos Hamiltonianos.

Por supuesto el problema ¿$\mathbb{P}$=$\mathbb{NP}$? se resolvería demostrando que un problema $\mathbb{NP}$-completo es $\mathbb{P}$ ó no es $\mathbb{NP}$.

1.13 Algoritmos probabilísticos

Un algoritmo se dice que resuelve un problema de decisión en *tiempo probabilístico polinomial* si es de tiempo polinomial y uno de los paso del algoritmo hace una elección aleatoria de un dato de forma que se cumplen las siguientes condiciones:

- Si la salida del algoritmo es positiva, entonces la respuesta al problema es positiva.

- Si la salida del algoritmo es negativa, entonces la probabilidad de que la respuesta verdadera sea negativa es mayor que 1/2.

Un *problema resoluble en tiempo probabilístico polinomial* es un problema para el que existe un algoritmo que lo resuelve en tiempo probabilístico polinomial.

Ejemplo 1.27 *Problema resoluble en tiempo polinomial probabilístico.*
Consideremos el problema de determinar si un polinomio $f \in \mathbb{Z}[X_1, \ldots, X_n]$ no es el polinomio nulo. Obviamente, este es un problema trivial si conocemos los coeficientes de f, pero supongamos que no estamos en condiciones de obtener estos coeficientes de forma eficiente, pero sí que sabemos evaluar $f(a_1, \ldots, a_n)$ en tiempo polinomial para todo $a_1, \ldots, a_n \in \mathbb{Z}$. Por ejemplo, esta situación se dará si f es la diferencia de los polinomios característicos de dos matrices de cuadradas orden 100.

Para resolver este problema utilizamos el siguiente algoritmo en el que se ha elegido un conjunto finito de números enteros I de forma que $|I| > cm$, donde m es el grado de f y $c > 2$. El algoritmo elige al azar un elemento $(a_1, \ldots, a_n) \in I^n$, calcula $f(a_1, \ldots, a_n)$. Si el resultado es $\neq 0$, decide afirmativamente sobre el problema, es decir, el polinomio no es nulo y, en caso contrario, decide que el polinomio es nulo.

Obsérvese que si la salida es positiva, el polinomio es efectivamente no nulo. Vamos a ver cuál es la probabilidad de que el polinomio sea no nulo y hayamos decidido erróneamente que si que lo es. Supongamos que $f \neq 0$. Por el lema que viene a continuación el número de ceros de f en I^n es menor o igual que $\frac{|I|^n}{c}$. Por tanto la probabilidad de que hayamos dado con una de dichas raíces es $\leq \frac{1}{c} < \frac{1}{2}$. Es decir la probabilidad de que el polinomio sea nulo es $> \frac{1}{2}$.

Lema 1.28 *Sea f un polinomio no nulo de grado m en n indeterminadas y coeficientes en un cuerpo K y sea I un subconjunto finito de K con más de cm elementos, con $c \geq 2$. Entonces el número de ceros de f en I^n es menor o igual que $\frac{|I|^n}{c}$.*

Demostración. Hacemos una doble inducción en n y m. Si $n = 1$, entonces f tiene a los sumo $m < \frac{|I|}{c}$ raíces. Supongamos que $n > 1$ y la hipótesis de inducción para menos de n variables. Por esta hipótesis de inducción podemos suponer sin pérdida de generalidad que todas las variables aparecen en f. Si $m = 1$, entonces para cada $(a_1, \ldots, a_{n-1})$, existe un único a_n tal que $(a_1, \ldots, a_{n-1}, a_n)$ es una raíz. Por tanto, el número de ceros de f es $\leq |I^{n-1}|$. Como $|I| \geq c$, necesariamente $|I|^{n-1} \leq \frac{|I|^n}{c}$. Supongamos ahora que $m \geq 2$ y la hipótesis de inducción para grado menor que m. Por reducción al absurdo, suponemos que f tiene más de $\frac{|I|^n}{c}$ ceros en I^n. Entonces para algún $a_n \in I$ existen más de $\frac{|I|^{n-1}}{c}$ elementos $(a_1, \ldots, a_{n-1}) \in I^{n-1}$ tales que $f(a_1, \ldots, a_{n-1}, a_n) = 0$. Eso implica que el polinomio $g(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, a_n)$ tiene más de $\frac{|I|^{n-1}}{c}$ ceros en I^{n-1}. Por hipótesis de inducción $g = 0$, o sea $f = (X_n - a_n)h$ para algún polinomio h. Entonces el grado de h es $m - 1$ y

$$|I| \geq cm = d(m-1), \text{ donde } d = c\frac{m}{m-1}.$$

Por hipótesis de inducción, h tiene a lo sumo $\frac{|I|^n}{d}$ ceros en $|I|^n$. Por tanto

$$\begin{array}{rcl} \frac{|I|^n}{c} & < & \text{N}^o \text{ de ceros de } f \text{ en } |I|^n \\ & \leq & |I|^{n-1} + \text{N}^o \text{ de ceros de } h \text{ en } |I|^n \\ & \leq & |I|^{n-1} + \frac{|I|^n}{d}. \end{array}$$

Dividiendo por $|I|^{n-1}$ y multiplicando por c obtenemos

$$|I| < c + \frac{|I|(m-1)}{m} = c + |I| - \frac{|I|}{m}$$

o sea

$$|I| < cm$$

en contra de la hipótesis. ∎

La idea de los algoritmos con tiempo polinomial aleatorio es que repitiendo su ejecución con distintas elecciones aleatorias, la certeza de obtener una solución correcta converge a 1.

1.14 Funciones de dirección única

Obsérvese que hemos utilizado un concepto no definido: función de dirección única.

Funciones de dirección única

De forma poco precisa diremos que una aplicación $f : A \to B$ se dice que es una *función de dirección única* si

- Para todo $a \in A$, se conoce cómo calcular $f(a)$ en un tiempo razonable (tiempo polinomial con exponente pequeño).
- Para la mayoría de los elementos b en la imagen de f, es difícil encontrar un $a \in A$ tal que $f(a) = b$ (problema $\mathbb{NP}$-completo).

La ambigüedad de la definición no depende solamente de la ambigüedad de las nociones de fácil y difícil sino también de las herramientas que se dispongan en cada momento tanto de hardware como de los algoritmos conocidos para resolver los problemas.

Ejemplos 1.29 *Funciones de dirección única*

(1) Sea $f \in K[X_1, \ldots, X_n]$ un polinomio con coeficientes en un cuerpo K y abusemos de la notación denotando también por f la función polinómica asociada $f : K^n \to K$. Esta es otra función de dirección única, salvo para polinomios de grado bajo, ya que es fácil calcular $f(a_1, \ldots, a_n)$ pero difícil resolver una ecuación del tipo $f(x_1, \ldots, x_n) = b$.

Más generalmente si $f_1, \ldots, f_m$ son polinomios de $K[X_1, \ldots, X_n]$ de grado mayor que 1 entonces la aplicación

$$\begin{array}{rcl} f : K^n & \to & K^m \\ x & \mapsto & (f_1(x), \ldots, f_m(x)) \end{array}$$

es habitualmente de dirección única. Esta es la base criptográfica de los esquemas de Lucifer.

(2) Denotamos por $\mathbb{P}$ al conjunto de números primos. La función

$$\begin{array}{rcl} f : \mathbb{P} \times \mathbb{P} & \to & \mathbb{Z} \\ (x, y) & \mapsto & xy \end{array}$$

es hoy por hoy una función de dirección única ya que aunque es fácil multiplicar dos números primos p y q, si lo que conocemos es su producto $n = pq$, descubrir los valores de p y q puede ser una tarea muy costosa computacionalmente.

(3) Consideremos dos números naturales m y n y consideremos la aplicación

$$\begin{array}{rcl} \mathbb{Z}_m & \to & \mathbb{Z}_m \\ x & \mapsto & x^n \end{array}$$

Esta función se puede calcular rápidamente con el método de doblar cuadrados que veremos más adelante. Sin embargo el problema de calcular la anti-imagen de un elemento por dicha aplicación, conocido como el *Problema del Logaritmo Discreto*, es un problema considerado difícil.

Más generalmente, supongamos que S es un semigrupo en el que la multiplicación en S se puede realizar en "tiempo rápido". Entonces doblando cuadrados, se puede calcular "rápidamente" la imagen de cualquier elemento de S por la siguiente función:

$$\begin{array}{rcl} S & \to & S \\ x & \mapsto & x^n \end{array}$$

El *Problema del Logaritmo Discreto* en S consiste en resolver en S ecuaciones del tipo $X^n = s$, con n un número natural y $s \in S$. Por tanto, la función anterior es de dirección única si el Problema del Logaritmo Discreto en S es un problema difícil.

Sin embargo hay semigrupos donde el Problema del Logaritmo Discreto se puede resolver en "tiempo rápido" como vamos a ver en la siguiente sección.

Recordemos que la bondad de un criptosistema se mide en términos de lo rápido que se pueden completar los procesos de cifrado y descifrado conociendo la clave y lo difícil que resulta sin conocerla. Por tanto, un buen criptosistema debe verificar que para casi toda clave k las aplicaciones $m \mapsto c(m, k)$ y $c \mapsto d(c, k)$ sean funciones de dirección única. Pero estamos hablando de tiempo de cálculo sin que hayamos definido este concepto de forma rigurosa. Ese es el objetivo de la siguiente sección.

1.15 Tareas

(1) Utilizamos un protocolo criptográfico afín en el que el alfabeto utilizado es el de las letras mayúsculas del castellano identificadas con los elementos de $\mathbb{Z}_{27}$ de forma que tres letras consecutivas cuyos números correspondientes forman el vector $v = (x, y, z)$ se han cifrado como $Av + b$, para $A \in \mathrm{GL}_3(\mathbb{Z}_{27})$ y $b \in \mathbb{Z}_{27}^3$. Sabiendo que con una clave (A, b) el mensaje "SIGUELAYLACONSEGUIRAS" se ha cifrado como

$$(0, 7, 10), (7, 11, 3), (13, 2, 12), (17, 6, 1), (19, 21, 14), (16, 22, 24), (12, 22, 26),$$

descifrar el siguiente mensaje cifrado que se ha encriptado con la misma clave:

$$(24, 7, 21), (24, 26, 23), (15, 11, 10), (8, 7, 21), (9, 19, 1), (9, 18, 10),$$
$$(21, 2, 14), (12, 19, 14), (7, 16, 8), (3, 13, 1), (23, 12, 20), (25, 20, 20), (8, 6, 26).$$

(2) Escribir un programa para calcular la entropía de los símbolos que aparecen en un texto y otro que utilizando el concepto de entropía calcule una estimación de la longitud de la clave de un texto cifrado con el criptosistema de Vigenère.

(3) Demostrar que las siguientes condiciones son equivalentes para dos variables aleatorias $\mathcal{M}$ y $\mathcal{C}$.

- $P(\mathcal{M} = m, \mathcal{C} = c) = P(\mathcal{M} = m)P(\mathcal{C} = c)$.
- $P(\mathcal{M} = m | \mathcal{C} = c) = P(\mathcal{M} = m)$.
- $P(\mathcal{C} = C | \mathcal{M} = m) = P(\mathcal{C} = c)$.

(4) Denotamos por $l(n)$ la longitud de un entero positivo escrito en una base fijada (indeterminada para este ejercicio). Demostrar las siguientes fórmulas:

(a) $l(n + m) \leq \max(l(n), l(m)) + 1$

(b) $l(nm) \leq l(n) + l(m)$

(5) Calcular una estimación del tiempo necesario para realizar los siguientes cálculos con números enteros positivos:

(a) División entera.

(b) Calcular el mínimo común múltiplo.

(c) Potenciación.

(d) Factorial.

(e) Expresar un número en una base.

(f) Determinar si un número es primo (utilizando la definición).

(g) Calcular el número de primos menores que el número dado.

(h) Factorizar un número en producto de primos.

(6) Escribir un programa que calcule el máximo común divisor de dos números enteros a y b utilizando el Algoritmo de Euclides, otro que encuentre una solución de la ecuación diofántica

$$aX + bY = \text{mcd}(a, b)$$

y otro que encuentre la solución de una ecuación diofántica del tipo

$$aX + bY = c.$$

(7) Implementa en GAP la exponenciación doblando cuadrados en un programa informático. Utilizalo para calcular potencias en $\mathbb{Z}_n$ y en cuerpos finitos.

(8) Estimar el tiempo cálculo de suma, producto, inversos y potencias en un cuerpo finito $\mathbb{F}_q$ con q elementos.

Indicación: Supongamos que $q = p^n$, con p primo. Suponer que se ha elegido un polinomio irreducible P en $\mathbb{F}_p = \mathbb{Z}_p$ de grado n, identificar $\mathbb{F}_q$ con $\mathbb{F}_p[X]/(P)$ e interpretar los elementos del cuerpo $\mathbb{F}_q$ como polinomios de grado menor que n. Para el cálculo del inverso adaptar el Algoritmo de Euclides Extendido a esta situación.

(9) Construir los cuerpos de 4, 8, 16, 9, 27 y 25 elementos como extensiones algebraicas de sus respectivos cuerpos primos. Calcular elementos primitivos de cada uno de ellos.

(10) Demostrar que si F es un cuerpo de característica $p > 0$ y n es un entero positivo, entonces existe una raíz n-ésima primitiva de la unidad en una extensión de F si y sólo si p no divide a n.

(11) Sea F una clausura algebraica de $\mathbb{F}_p$ y sea $\alpha \in F$. Denotamos por $\mathbb{F}_{p^n}$ al único subcuerpo de F con p^n elementos. Demostrar las siguientes propiedades para $\alpha \in F$.

(a) α es una raíz de la unidad.

(b) $\alpha \in \mathbb{F}_{p^m}$ si y sólo si el orden de α divide a $p^m - 1$.

(c) Supongamos que α tienen orden n y $p^k = |F_p(\alpha)|$. Entonces k es el orden de p en el grupo de unidades de $\mathbb{Z}_n$, es decir k es el menor entero positivo k para el que $p^k \equiv 1 \mod n$.

(d) La aplicación $\sigma : x \to x^p$ es un automorfismo de F (y de cada $\mathbb{F}_{p^m}$). Este automorfismo se llama el *automorfismo de Frobenius*.

(e) Si $q = p^k$ entonces la restricción de σ a $\mathbb{F}_q$ tiene orden k. Usar esto para deducir que para todo entero positivo m, el cuerpo $\mathbb{F}_{q^m}$ es una extensión de Galois de $\mathbb{F}_q$ de orden m y su grupo de Galois es cíclico generado por σ^k.

(12) En este ejercicio vamos a marcar los pasos para poder construir programas de cifrado usando GAP. Vamos a usar como alfabeto los símbolos de la función `CharInt` cuya inversa es la aplicación `IntChar`.

(a) Comienza familiarizandote con las dos funciones calculando `CharInt(i)` para varios valores i entre 0 y 255 y al contrario calcula `IntChar('a')` para varios símbolos `a`. Descubriras que hay algunos símbolos como la ñ o las letras acentuadas que no puedes escribir en el prompt de GAP. Para poder hacerlo tendrás que escribirlos en un fichero de texto y leerlos usando la función `Read`. El símbolo que nunca podrás usar es `"` pues significa principio o final de texto.

(b) Convierte varios textos en listas de números como en el siguiente ejemplo:

```
gap> List("Al andar se hace camino",IntChar);
[ 65, 108, 32, 97, 110, 100, 97, 114, 32, 115, 101, 32, 104, 97, 99, 101,
32, 99, 97, 109, 105, 110, 111 ]
```

Para poder usar acentos y ñ escribe los textos en un fichero de texto y leelos usando `Read`.

(c) Haz lo contrario del apartado anterior, es decir convierte en textos varias listas con entradas en $\mathbb{Z}_{256}$.

(d) Construye una función de GAP que para cualquier entero positivo n implimente la biyección y otro para implementar su inversa:

$$\begin{array}{rcl} \mathbb{Z}_{256}^n & \to & \mathbb{Z}_{256^n} \\ (x_0, x_1, \ldots, x_{n-1}) & \mapsto & x_0 + x_1 256 + x_2 256^2 + \cdots + x_{n-1} 256^{n-1} \end{array}$$

(e) Construye una función de GAP que dado un texto y un entero positivo n convierta el texto en una lista de elementos de $\mathbb{Z}_{256^n}$ y otro que haga lo contrario combinando lo que has aprendido en los apartados (b) y (c) con los programas del apartado (d).

(f) Usa las funciones anteriores para escribir implementaciones de los criptosistemas que hemos visto en este capítulo. Estos programas deben permitir elegir como parámetro cual es la plataforma $\mathbb{Z}_{256^n}$ en la que se representas los textos en claro y cifrados.

(13) Escribir un programa para cifrado y descifrado con un esquema de Lucifer binario utilizando la función polinómica

$$f(x_1, x_2, x_3; y_1, y_2, y_3) = (x_1 x_2 y_1 y_2, x_1 x_3 y_1 y_3, (x_1 + x_2 + x_3)(y_1 + y_2 + y_3),$$

de forma que $M = C = \mathbb{Z}_2^6$, la clave inicial es una palabra

$$\mathbf{k} = (k_1, k_2, \ldots, k_7)$$

de longitud 7 y realizamos tres iteraciones de cifrado con las claves

$$\mathbf{k}_1 = (k_2, k_4, k_6), \quad \mathbf{k}_2 = (k_1, k_3, k_5), \quad \mathbf{k}_3 = (k_7, k_2, k_4), \quad \mathbf{k}_4 = (k_5, k_7, k_3)$$

Asignar a los $2^6 = 64$ mensajes básicos los siguientes símbolos: Las 27 letras mayúsculas, las 27 minúsculas del alfabeto español, el espacio en blanco y los siguientes 9 símbolos: ,;.!¡¿?(). Completar el programa para que cifre y descifre mensajes escritos con estos símbolos.

Capítulo 2

Criptosistemas asimétricos o de clave pública

Los criptosistemas simétricos se llaman así porque la información necesaria para cifrar es esencialmente la misma que se precisa para descifrar. El crecimiento de las telecomunicaciones ha mostrado la necesidad de la comunicación segura entre partes que tal vez no se han encontrado ni se vayan a encontrar nunca. Esto hace necesario contar con criptosistemas en los que la información necesaria para cifrar no sea suficiente para descifrar. Estos criptosistemas se suelen llamar asimétricos o de clave pública. Por otro lado, uno de los problemas fundamentales de la criptografía es el de acordar una clave entre dos partes que pueden no tener una comunicación segura. Esta es una situación habitual pues a menudo se utilizan criptosistemas simétricos, por ser normalmente más rápidos que los asimétricos, y es necesario ponerse de acuerdo en la clave usando un canal de comunicación al que puede tener acceso el adversario. Esto dio lugar a la idea de la clave pública.

2.1 Intercambio de claves de Diffie-Hellman

El primer protocolo de intercambio de claves que usa la idea de la clave pública fue propuesto en 1976 por Diffie y Helman y funciona de la siguiente forma. Supongamos que dos usuarios, Alicia y Blas, quieren ponerse de acuerdo en una clave. Lo primero que hacen es elegir dos números naturales coprimos n y g de forma que a cada elemento del conjunto $\langle g\rangle_n$ formado por las potencias de g en $\mathbb{Z}_n$ se le puede asociar una de las claves de nuestro sistema. Es decir, podemos considerar que el conjunto de claves del criptosistema simétrico que utilizan Alicia y Blas para su transmisión habitual es $\langle g\rangle_n = \{1, g, g^2, g^3, \dots\}$ (aunque en realidad basta con que haya una aplicación de $\langle g\rangle_n$ al conjunto de claves). Tanto n como g pueden ser públicos, y de hecho, puede ser que no sólo Alicia y Blas sean los usuarios de este sistema sino que estemos hablando de un protocolo criptográfico simétrico establecido como un estándar de comunicación en el que las claves están codificadas por los elementos de $\langle g\rangle_n$. Para que Alicia y Blas se pongan de acuerdo en una clave, cada uno elige un número. Pongamos que Alicia elige el número x_A y Blas elige el número x_B. Entonces cada uno eleva g al número elegido y envía el resultado al otro. Es decir, Alicia calcula $k_A = (g^{x_A} \mod n)$ y envía este resultado a

Blas y Blas calcula $k_B = (g^{x_B} \mod n)$ y envía k_B a Alicia. Entonces Alicia, que ha recibido k_B de Blas y conoce n_A, calcula $(k_B^{n_a} \mod n)$ y Blas, que ha recibido k_A de Alicia y conoce n_B, calcula $k_A^{n_B}$. El siguiente esquema muestra los cálculos realizados por Alicia y Blas y la información trasmitida. Lo que aparece en la columna central es conocido tanto por Alicia y Blas, como por cualquiera que sea capaz de interceptar la comunicación entre ellos. Lo que aparece en la columna izquierda sólo es conocido por Alicia y lo que aparece en la columna derecha sólo es conocido por Blas.

Alicia			Público		Blas		
n_A	$\rightarrow$	$g^{n_A} =$	k_A	k_B	$= g^{n_B}$	$\leftarrow$	n_B
$\downarrow$							$\downarrow$
$k_B^{n_A}$	$= g^{n_B n_A}$		g	n		$g^{n_A n_B} =$	$k_A^{n_B}$

El resultado obtenido por Alicia y Blas coincide pues

$$k_B^{n_A} = (g^{n_B})^{n_A} = g^{n_A n_B} = (g^{n_A})^{n_B} = k_A^{n_B}.$$

Por tanto, Alicia y Blas pueden fijar como clave para su comunicación el resultado obtenido independientemente por ambos, es decir $k = g^{n_A n_B}$.

El protocolo de intercambio de claves de Diffie-Helman puede generalizarse de la siguiente forma: Partimos de una aplicación

$$f : K \times P \rightarrow K$$

que cumpla las siguientes propiedades:

(1) Para todo $k \in K$ y $p_1, p_2 \in P$ se verifica la siguiente igualdad :

$$f(f(k, p_1), p_2) = f(f(k, p_2), p_1). \tag{2.1}$$

(2) Para la mayoría de los elementos p de P, la aplicación $k \mapsto f(k, p)$ es de dirección única.

El conjunto K es el conjunto de las claves posibles para el criptosistema simétrico utilizado por dos usuarios que quieren ponerse de acuerdo en una clave. Los elementos de P van a ser parámetros que van a ayudar a ponerse de acuerdo en la clave. El procedimiento de establecimiento de clave común es el siguiente. Los dos usuarios se ponen de acuerdo en un valor s de K, que va a servir de *semilla* de la clave. No importa que s sea descubierto por el adversario por lo que se lo pueden comunicar en claro. Cada uno de los usuarios elige en secreto un parámetro de P. Pongamos que Alicia elige $p_A \in P$ y Blas elige $p_B \in P$. Entonces Alicia calcula $k_A = f(s, p_A)$ y Blas calcula $k_B = f(s, p_B)$. Ambos usuarios se transmiten los valores k_A y k_B. Con el valor k_B que Alicia ha recibido de Blas, aquella puede calcular

$$k = f(k_B, p_A)$$

y con el valor k_A que Blas ha recibido de Alicia, Blas puede calcular

$$k' = f(k_A, p_B).$$

Pero

$$k' = f(k_A, p_B) = f(f(s, p_A), p_B) = f(f(s, p_B), p_A) = f(k_B, p_A) = k,$$

es decir los dos usuarios han obtenido el mismo valor que pueden utilizar como clave a partir de ese momento. El esquema es similar al de Diffie-Hellman:

Alicia			Público		Blas		
p_A	$\rightarrow$	$f(s, p_A) =$	k_A	k_B	$= f(s, p_B)$	$\leftarrow$	p_B
$\downarrow$							$\downarrow$
$f(k_B, p_A)$	$= k$		s	f		$k' =$	$f(k_A, p_B)$

Obsérvese que aunque un intruso conozca la función f y logre interceptar los mensajes enviados: s, k_A, k_B, no le resultará fácil calcular $k = k'$, ya que para ello necesita, en teoría, conocer p_A ó p_B. Para que esta afirmación sea correcta la aplicación $p \mapsto f(s, p)$ debe ser una función de dirección única, aunque puede ser que esto no sea suficiente pues el adversario conoce dos valores k_A y k_B de P y le bastaría con resolver una de las dos ecuaciones $f(s, X) = k_A$ ó $f(s, X) = k_B$.

2.2 Criptosistemas asimétricos o de clave pública

Definición 2.1 *Un* criptosistema de clave pública *o* criptosistema asimétrico *está formado por:*

- *Una función* $P : K' \rightarrow K$*;*
- *para cada* $k \in K$ *una aplicación* $c_k : M_k \rightarrow C_k$ *y*
- *para cada* $k' \in K'$ *una aplicación* $d_{k'} : C_{k'} \rightarrow M_{k'}$

de forma que si $k = P(k')$ *entonces*

$$M_k = M_{k'}, \quad C_k = C_{k'}, \quad y \quad d_{k'} c_k(m) = m, \quad \text{para todo } m \in M_k.$$

Un usuario de un criptosistema de clave pública elegirá $k' \in K'$, calculará $k = P(k')$ y hará público k. Por esto k se llama *clave pública* mientras que k' se llama *clave privada.* Los otros usuarios podrán enviarle mensaje cifrados utilizando la función de cifrado $c_k : M_k \rightarrow C_k$. Para descifrar el mensaje cifrado $c = c_k(m)$ se usará la clave privada k' para calcular $d_{k'}(c) = d_{k'} c_k(m) = m$.

Aparte de que se cumpla $d'_k \circ c_k = 1_M$, para que un criptosistema de clave privada sea seguro es necesario que se cumplan las siguientes condiciones:

(1) P es una función de dirección única.

(2) Para la mayoría de los $k \in K$ la aplicación c_k es una función de dirección única.

(3) $d_{k'}$ se puede calcular en tiempo rápido si se conoce la clave privada k' pero difícil sin conocerla, aunque se conozca la clave pública k.

La idea de los criptosistemas de clave pública es poder establecer comunicación privada segura entre varios usuarios que tal vez no tienen nunca contacto privado entre ellos, de forma que cada uno pueda enviar mensajes cifrados a todos los demás, pero sólo pueda descifrarlos el receptor legítimo del mensaje. Imaginemos, por ejemplo, las sucursales de un banco repartidas por el mundo. Entre ellas necesitan hacer transacciones de forma privada. Se establece un criptosistema de clave pública y la central del banco asigna a cada sucursal una clave que hace pública de alguna forma. Para ello asigna a cada sucursal un elemento $k' \in K'$, o sea su clave privada, y calcula las claves públicas $P(k')$ de todas las sucursales. Cada sucursal puede tener un fichero con las claves públicas de todas las sucursales pero la clave privada de una sucursal sólo es accesible para las personas autorizadas de esa sucursal. Cuando la sucursal A quiere mandar un mensaje cifrado a la sucursal B, busca en su fichero cuál es la clave pública de la sucursal B, pongamos k_B y la utiliza para cifrar. Es decir, para enviar a la sucursal B el mensaje en claro m, éste se cifra como

$$c = c_{k_B}(m).$$

Cuando la sucursal B reciba el mensaje cifrado c, lo descifrará utilizando su clave privada k'_B, que recordemos que sólo conocen las personas autorizadas de dicha sucursal. Para ello calcula

$$d_{k'_B}(c) = d_{k'_B}(c_{k_B}(m)) = m.$$

Obsérvese que, de acuerdo a los requerimientos anteriores, el conocimiento de la clave privada k'_B permite calcular $c_{k'_B}(c)$ de forma rápida. Sin embargo, el conocimiento de $c = d_{k_B}(m)$ y de la clave pública k_B, no debe de ser suficiente para calcular m sin conocer la clave privada, ya que la aplicación c_{k_B} debe de ser una función de dirección única.

La criptografía asimétrica tienen otras aplicaciones, aparte de la de transmisión de información cifrada entre muchos usuarios por canales no seguros. Veamos algunos de ellos.

Firma digital

La *firma digital* es un sistema para poder identificar al autor de un documento. Por ejemplo, podemos necesitar estar seguros de que el cliente que nos ha hecho un pedido o el usuario de un cajero automático es quien dice ser. Por ejemplo, necesitamos esta información para saber si podemos fiarnos del cliente o para poder probar delante de un tribunal que él nos ha hecho el pedido o realizado el reintegro. Para poder utilizar un criptosistema con estos fines es necesario que las aplicaciones c_k y $d_{k'}$, con k y k' las claves públicas y privadas del cliente, sean mutuamente inversas, es decir, aparte de las condiciones anteriores se debe verificar

$$c_k(d_{k'}(c)) = c, \quad \text{para todo } k' \in K, k = P(k) \text{ y } c \in C_k.$$

Supongamos que Alicia y Blas, tienen claves públicas k_A y k_B y claves privadas k'_A y k'_B. La firma en claro de Alicia es un cierto texto en claro F_A y esta firma es pública. Entonces la firma digital de A para identificarse delante de B es

$$F_{A,B} = c_{k_B} d_{k'_A}(F_A).$$

Obsérvese que Alicia puede calcular $F_{A,B}$ pues conoce su propia clave privada k'_A y la clave pública de Blas, k_B. El hecho de que la firma en claro sea pública no le quita valor como identificador de Alicia ya que la que verdaderamente vale es $F_{A,B}$ que sólo puede ser calculada usando k'_A, que se supone que sólo conoce Alicia. Cuando Alicia quiere identificarse delante de Blas, envía $F = F_{A,B}$. Entonces Blas puede utilizar F para comprobar que Alicia es quien dice ser calculando $c_{k_A} d_{k'_B}(F)$ y comprobando si coincide con F_A, como debería ocurrir si Alicia ha usado k'_A pues

$$c_{k_A} d_{k'_B}(F) = c_{k_A} d_{k'_B}(c_{k_B}(d_{k'_A}(F_A)) = c_{k_A} d_{k'_A}(F_A) = F_A.$$

Blas puede hacer esta operación, es decir calcular $c_{k_A} d_{k'_B}(F)$, pues conoce la clave pública k_A de Alicia y su propia clave privada k'_B.

Aparte de la utilidad para tener seguridad de que el autor del mensaje es quien dice ser, la firma digital tiene la ventaja de servir de prueba contra el *repudio* del autor del mensaje. Esto es lo que se conoce como *no repudio*.

Integridad

Supongamos que queremos estar seguros de que un documento no ha sido alterado. Por ejemplo, supongamos que queremos estar seguros de que en el expediente académico que ha sido entregado por un alumno de la facultad de matemáticas para obtener un trabajo, no se ha modificado ninguna calificación. Este expediente va firmado por el secretario de la facultad, e incluso podemos tener un sistema de firma digital, lo que nos garantiza que la firma es auténtica, pero eso no garantiza que el contenido no haya sido modificado. Otro ejemplo, bastante común es el de saber que no se ha añadido un virus a un archivo. La característica de que no se haya modificado un documento se llama *integridad*. Para garantizar la integridad se utilizan una *función de resumen* que en la literatura en inglés se denomina *hash function*[1].

Una *función de resumen* ó *función hash* es una aplicación

$$H : M \to R$$

que satisface:

- Se puede calcular $H(m)$ de forma rápida (tiempo polinomial de exponente bajo).
- Aunque H no sea inyectiva, lo es a efectos prácticos, en el sentido de que dado $m \in M$ es difícil encontrar $m' \neq m$ tal que $H(m') = H(m)$.

Existen funciones resumen estandarizadas (SHA, MD4) que el lector puede encontrar fácilmente en la literatura tanto en libros como en internet. Nosotros no vamos a entrar en la naturaleza de ningún ejemplo de función resumen. Simplemente vamos a ver cómo se pueden utilizar para garantizar la integridad de un documento.

Si aplicamos una función resumen H a un mensaje, obtendremos un valor $r = H(m)$, que llamaremos *resumen del mensaje*. Ahora podemos considerar este resumen como una firma en claro y calcular la firma digital asociada. Por ejemplo, si es Alicia la que envía el mensaje a Blas, entonces calculará

$$RF = c_{k_B} d_{k'_A}(r)$$

[1] hash=picar y mezclar

y lo adjuntará al mensaje. Llamaremos a RF *resumen firmado*. Cuando Blas recibe de Alicia el mensaje m con su resumen firmado RF, puede comprobar de una tanto que el mensaje no ha sido alterado como que se trata de un mensaje de Alicia. Para ello aplica la función resumen H al mensaje y obtiene el resumen en claro $r = H(m)$. (Recordemos que la función resumen es un estándar en el que los dos usuarios se han puesto de acuerdo.) Después, comprueba la firma calculando

$$c_{k_A} d_{k'_B}(RF)$$

que debe coincidir con r si el mensaje no ha sido alterado y la firma es del emisario legítimo, es decir de Alicia.

Palabras de acceso (passwords)

Una última aplicación de los criptosistemas de clave pública es el del control de palabras de acceso a un sistema (*passwords*). Por ejemplo, supongamos que administramos un superordenador utilizado por muchos usuarios en el que el tiempo de acceso o los recursos están limitados o que los distintos usuarios tienen acceso a diferentes recursos del ordenador. Para evitar que intrusos puedan utilizar el tiempo o los recursos del ordenador a los que no estén autorizados se le asigna a cada usuario una palabra clave. Para que el ordenador pueda reconocer la palabra clave necesita tenerla almacenada en algún lado. Eso lleva consigo el peligro de que un usuario logre descubrir dónde tiene el ordenador almacenadas las palabras de acceso y pueda utilizar palabras de acceso de otros usuarios. Un sistema de seguridad consiste en almacenar las palabras de acceso cifradas. Por ejemplo, para un usuario el ordenador almacena su clave pública k y su la clave de acceso en forma cifrada P'. Es decir, si P es la palabra de acceso de dicho usuario entonces lo que se almacena además de k es $P' = c_k(P)$. Cuando el usuario desea acceder a los recursos del ordenador necesita teclear P. El ordenador calcula $c_k(P)$ y comprueba que coincide con el valor P' que tiene almacenado.

Criptosistemas simétricos que pueden parecer asimétricos

Podría parecer que algunos de los criptosistemas simétricos vistos en el Capítulo 1 podrían ser considerados como de clave pública. Por ejemplo, podíamos interpretar el Criptosistema de César como de clave pública tomando $K = K' = \mathbb{Z}_{27}$ y considerando como función que relaciona las claves privadas y públicas $P(k) = (-k \mod 27) = 27 - k$ y como función de cifrado $c_k(x) = (x + k \mod 27)$ y de descifrado $d_{k'} = c_{P(k)}$. Sin embargo, esta función no puede ser de dirección única pues $P = P^{-1}$ y, por tanto, es tan difícil calcular $P(k)$ como $P^{-1}(k')$. Otro ejemplo, un poco menos obvio es el de un criptosistema afín. Recordemos que en tal caso $M = C = \mathbb{Z}_n$, $K = \mathbb{Z}_n^* \times \mathbb{Z}_n$ y las funciones de cifrado y descifrado son las de (1.2). Si ponemos $K' = K$, $P(x, y) = (x^{-1}, -x^{-1}y)$ y $c_{k'} = c_k$ tendríamos que se cumplirían las hipótesis de un criptosistema asimétrico. Sin embargo tampoco P es una función de dirección única pues es fácil calcular a^{-1} a partir de a, usando el Algoritmo de Euclides Extendido, como vimos en la Sección 1.8, y a partir de ahí es fácil calcular la clave privada $k' = (a^{-1}, -a^{-1}b)$ a partir de la clave pública $k = (a, b)$.

2.3 RSA

Ya va siendo hora de que veamos algún criptosistema asimétrico concreto.

El criptosistema de clave pública más popular es el conocido como RSA, por las iniciales de los autores que lo patentaron en el año 1978: Rivest, Shamir y Adleman Está basado en el Teorema de Euler:

$$\text{Si } \mathrm{mcd}(x,n)=1, \text{ entonces } x^{\phi(n)} \equiv 1 \mod n.$$

donde ϕ representa la función de Euler, es decir

$$\phi(n) = |\mathbb{Z}_n^*|.$$

Obsérvese que el Teorema de Euler tiene la siguiente consecuencia:

$$\text{Si } x,n \in \mathbb{Z}, \gcd(x,n)=1 \text{ y } m \equiv 1 \mod \phi(n) \text{ entonces } x^m \equiv x \mod n. \tag{2.2}$$

Recordemos que ϕ se puede calcular con la siguiente fórmula:

$$\phi(n) = n \prod_{p|n, p \text{ primo}} \left(1 - \frac{1}{p}\right).$$

En particular, si $n = pq$, donde p y q son dos primos distintos, entonces $\phi(n) = (p-1)(q-1)$.

En realidad la verdadera base de RSA es la siguiente proposición:

Proposición 2.2 *Sea $n = pq$, con p y q primos distintos. Si $x \equiv 1 \mod \phi(n)$, entonces $a^x \equiv a \mod n$, para todo $a \in \mathbb{Z}$.*

Demostración. Si a es múltiplo de n, entonces $a^x \equiv 0 \equiv a \mod n$. Si a y n son coprimos entonces de (2.2) tenemos que $a^x \equiv a \mod n$. En los casos que quedan a es múltiplo de p ó q pero no del otro. Por simetría, supondremos que a es múltiplo de p pero no de q. Entonces $a^x \equiv 0 \equiv a \mod p$ y $a^x \equiv a \mod q$ por el Teorema de Fermat. Como p y q son coprimos, usando el Teorema Chino de los Restos deducimos que $a^x \equiv a \mod n$. ∎

Descripción de RSA

El criptosistema RSA funciona de la siguiente forma. El usuario elige dos números dos primos distintos p y q, calcula $n = pq$ y $\phi(n) = (p-1)(q-1)$, elige un elemento $c \in \mathbb{Z}_{\phi(n)}^*$ y calcula el inverso d de c en $\mathbb{Z}_{\phi(n)}$. La clave pública de dicho usuario es la pareja $k = (n,c)$ y su clave privada es $k' = (n,d)$. El conjunto de los mensajes en claro y cifrados es $M_k = C_k = \mathbb{Z}_n$. Entonces las aplicaciones de cifrado y descifrado con las claves pública $k = (n,c)$ y privada $k' = (n,d)$ son

$$\begin{array}{rcl} c_k : \mathbb{Z}_n & \to & \mathbb{Z}_n \\ a & \mapsto & a^c \end{array} \qquad \begin{array}{rcl} d_{k'} : \mathbb{Z}_n & \to & \mathbb{Z}_n \\ a & \mapsto & a^d \end{array}$$

La seguridad de RSA depende del tamaño de los números primos p y q con respecto a la capacidad de cálculo del intruso. Hasta 2003 se consideraba seguro utilizar primos de forma

que $n = pq$ sea un número de $1024 = 2^{10}$ bits. Es decir, la longitud de n en binario debe de ser al menos de 1024 dígitos y en consecuencia las longitudes de p y q en binario deben de ser de unos 512 bits. En 2003 Shamir y Tromer mostraron un dispositivo de una máquina teórica que podría factorizar un número de 1024 bits[2]. En la actualidad se recomienda duplicar la longitud de los n, es decir utilizar primos para los que n tenga 2048 bits.

Veamos que es lo que nos permite considerar RSA como un criptosistema de clave pública. En primer lugar cumple el requerimiento mínimo de que el mensaje descifrado sea el correcto pues si $k = (n, c)$ y $k' = (n, d)$ entonces $cd \equiv 1 \mod \phi(n)$ (pues d es el inverso de c en $\mathbb{Z}^*_{\phi(n)}$). Entonces, haciendo cálculos en $\mathbb{Z}_n$ y utilizando la Proposición 2.2, tenemos

$$d_{k'} c_k(m) = (m^c)^d = m^{cd} = m.$$

En segundo lugar, necesitamos que para la mayoría de las claves públicas $k = (n, c) \in K$ la aplicación $c_k : a \mapsto (a^c \mod n)$ sea función de dirección única y que para toda clave privada $k' = (n, d)$ se pueda calcular $c_k : a \mapsto (a^d \mod n)$ de forma rápida. En efecto, las potencias módulo n se pueden calcular de forma rápida con el algoritmo de Exponenciación Doblando Cuadrados visto en la Sección 1.9. Sin embargo el problema de calcular inversos de esta función es el resolver una ecuación de congruencias del tipo

$$X^c \equiv a \mod n, \quad (a, c \in \mathbb{Z}, \gcd(c, \phi(n)) = 1),$$

que es un subproblema del problema de calcular raíces n-ésimas módulo n.

Finalmente necesitamos que la aplicación P que asocia claves públicas y privadas sea de dirección única. Tal como hemos descrito el criptosistema el conjunto de claves públicas y privadas es

$$K = \{(n, c) \in \mathbb{N}^2 : n \text{ es el producto de dos primos distintos tales que } \mathrm{mcd}(c, \phi(n)) = 1\}$$

y la aplicación P que relaciona las claves privadas con las públicas viene dada por $P(n, d) = (n, c)$, con $cd \equiv 1 \mod \phi(n)$. Para que se vea bien que esta función es de dirección única debemos modificar ligeramente las claves privadas. Obsérvese que aunque el usuario con clave privada (n, d), sólo utiliza para descifrar n y d, en realidad tiene más información, pues ha elegido previamente dos primos p y q con los que ha calculado $n = pq$ y $\phi(n) = (p-1)(q-1)$. Digamos pues que la clave privada es $(n, \phi(n), d)$. Entonces la aplicación que asocia claves privadas con públicas es

$$\begin{array}{rccc} P: & K' & \to & K \\ & (n, \phi(n), d) & \mapsto & (n, (d^{-1} \mod \phi(n))) \end{array}$$

donde K es como antes y

$$K' = \{(n, \phi(n), d) : (x, d) \in K\}.$$

La función P es una función de dirección única pues dado $(n, \phi(n), d) \in K$ podemos calcular fácilmente el inverso c de d módulo $\phi(n)$ con el algoritmo para calcular inversos modulares

[2]Shamir y Tromer, Factoring Numbers with TWIRL device, Proc. Crypto 2003,, LNCS 2729, 1–26, Springer-Verlag 2003

visto en la Sección 1.8. Sin embargo, dado $k = (n, c) \in K$ no resulta tan fácil calcular la anti-imagen por P de k. En realidad, para romper el criptosistema sólo tenemos que calcular d, el inverso de c en $\mathbb{Z}^*_{\phi(n)}$, lo que es fácil de hacer utilizando el algoritmo para calcular inversos modulares. Sin embargo, para poder utilizar este algoritmo necesitamos conocer $\phi(n)$, que no se ha hecho público. El valor de $\phi(n)$ se podría calcular a partir de la descomposición de n, pues si esta descomposición es $n = pq$, entonces $\phi(n) = (p-1)(q-1)$. Lo que resulta difícil es calcular $\phi(n) = (p-1)(q-1)$ sin conocer la factorización de n. Vamos a ver esto en la siguiente subsección.

Criptoanálisis de RSA

Vamos a dedicar el resto de la sección a analizar los métodos más naturales de criptoanalizar RSA. En la descripción del Criptosistema RSA hemos visto que en realidad no necesitamos toda la información de $P^{-1}(n, c)$ para romper el sistema pues para descifrar sólo utilizamos n y $d = (c^{-1} \mod \phi(n))$. Para calcular d basta con conocer $\phi(n)$ y aplicar el método para calcular inversos modulares explicado en la Sección 1.8. Es decir:

Proposición 2.3 *El problema de romper RSA se reduce en tiempo polinomial al problema de calcular $\phi(n)$ para n un producto de dos números primos.*

Ya sabemos que calcular $\phi(n)$ es fácil a partir de la factorización de n. Dicho en términos técnicos, el problema de calcular ϕ se reduce al problema de la factorización en tiempo polinomial. Por tanto, de la Proposición 2.3 se deduce el siguiente

Corolario 2.4 *El problema de romper RSA se reduce en tiempo polinomial al de factorizar un número que sea producto de dos números primos distintos.*

Se ha conjeturado que también es cierto el recíproco del Corolario 2.4, es decir que el problema de factorizar un número que sea producto de dos números primos es equivalente en tiempo polinomial al de romper RSA. De momento éste sigue siendo un problema abierto.

Teóricamente el problema de calcular ϕ podría ser más fácil que el de factorizar. Sin embargo la siguiente proposición muestra que, en realidad, son equivalentes en tiempo polinomial.

Proposición 2.5 *Dados $n = pq$, con p y q primos distintos, se pueden calcular p y q a partir de n y $\phi(n)$ en tiempo $O(\log^3 n)$.*

Demostración. La proposición es trivial si n es par pues en tal caso $p = 2$ y $q = n/2$ o viceversa. Supongamos que n es impar. Obsérvese que $\phi(n) = n - p - q + 1$, con lo que p y q son las soluciones de la ecuación de segundo grado

$$(X - p)(X - q) = X^2 - 2b + n$$

donde $2b = p + q = n - \phi(n) + 1$, es decir

$$p, q = b \pm \sqrt{b^2 - n}.$$

La única operación para la que todavía no hemos estimado el tiempo de cálculo es la raíz cuadrada. Lo que hacemos en el siguiente lema que nos dará el resultado deseado. ∎

Lema 2.6 *El tiempo necesario para calcular una raíz cuadrada de un número con k dígitos binarios es $O(k^3)$.*

Demostración. Sea x un número natural con k dígitos binarios, es decir

$$2^{k-1} \leq x < 2^k.$$

Entonces

$$2^{\frac{k-1}{2}} \leq \sqrt{x} < 2^{\frac{k}{2}}.$$

Luego $\sqrt{x}$ tiene longitud $l = \lceil \frac{k}{2} \rceil$. Por tanto, si $\sqrt{x} = y_{l-1} \dots y_1 y_0$ es la expresión binaria de $\sqrt{x}$ entonces $y_{l-1} = 1$ y

$$\sqrt{x} = y_0 + 2y_1 + 2^2 y_2 + \cdots + 2^{l-1} y_{l-1}, \quad \text{con} \quad y_i = \begin{cases} 0, & \text{si } (2^i y_i + 2^{i+1} y_{i+1} + \cdots + 2^{l-1} y_{l-1})^2 > x; \\ 1 & \text{en otro caso.} \end{cases}$$

Por tanto podemos calcular $\sqrt{x}$ con el siguiente algoritmo:

Algoritmo 6. Cálculo de raíces cuadradas

ENTRADA: x, un cuadrado perfecto.
$k := l(x)$, $l := \lceil \frac{k}{2} \rceil$, $y := 2^{l-1}$, $z = y$.
Mientras que $z > 1$,
$z := z/2$,
si $(y+z)^2 \leq x$, entonces $y = y + z$.
SALIDA: y.

El algoritmo ejecuta $l-1$ bucles y cada bucle esta formado por operaciones cuyo tiempo de ejecución es del orden $O(k^2)$. Por tanto el tiempo de calculo del algoritmo es $O(k^3)$. Dejamos como ejercicio comprobar que la salida y de este algoritmo cumple $y^2 = x$, si x es un cuadrado perfecto. ∎

Recordemos que el Teorema de Euler nos asegura que $m = \phi(n)$ satisface la siguiente propiedad:

$$x^m \equiv 1 \mod n, \text{ para todo } x \text{ con } \mathrm{mcd}(x, n) = 1. \tag{2.3}$$

Hemos visto cómo calcular la factorización de $n = pq$, con p y q primos distintos a partir de $\phi(n)$. Vamos a ver ahora un algoritmo probabilístico que también calcula la factorización de n en tiempo polinomial probabilístico a partir de un entero positivo m que satisfaga (2.3). Para ello necesitaremos el siguiente lema.

Lema 2.7 *Sea G un grupo abeliano y sea k un entero positivo. Si existe un $g \in G$ tal que $g^k \neq 1$, entonces el conjunto $\{x \in G : x^k = 1\}$ tiene cardinal menor o igual que $|G|/2$.*

Demostración. El conjunto $X = \{x \in G : x^k = 1\}$ es un subgrupo de G y por tanto su cardinal divide al de G. Luego, si $X \neq G$, entonces $|X| \leq |G|/2$. ∎

Teorema 2.8 *El problema de factorizar un número que es el producto de dos primos distintos se reduce en tiempo polinomial probabilístico al de encontrar un entero m que verifique la propiedad (2.3).*

Demostración. Sea n un número que es el producto de dos primos distintos. Está claro que podemos suponer que n es impar. Supongamos además que conocemos un número m que satisface (2.3). Para simplificar la notación vamos a trabajar en $\mathbb{Z}_n$, con lo que eliminamos todas las referencias a reducciones módulo n.

En primer lugar obsérvese que, como n es impar, m ha de ser par ya que

$$(-1)^{2t+1} = -1 \neq 1.$$

(Recuérdese que estamos trabajando en $\mathbb{Z}_n$, es decir la anterior fórmula $-1 \neq 1$ significa $-1 \not\equiv 1 \mod n$.)

La primera parte del algoritmo hace los siguiente:

Mientras que m sea par y $m/2$ satisfaga (2.3), hacemos $m := m/2$.

Para hacer esto efectivo necesitamos un algoritmo que decida si un número k satisface la propiedad (2.3) o no. En realidad, se trata del algoritmo probabilístico que simplemente elige un elemento de $\mathbb{Z}_n^*$ al azar y comprueba si $a^k \equiv 1 \mod n$. Para que esto sea efectivamente un algoritmo probabilístico necesitamos que si k no satisface la propiedad (2.3), entonces la probabilidad de que al elegir al azar un elemento de $\mathbb{Z}_n^*$, sí que se verifique $a^k \equiv 1 \mod n$, sea menor o igual que $1/2$. Esto es consecuencia del Lema 2.7. Por tanto, si m satisface la propiedad (2.3), elegimos varios elementos de $\mathbb{Z}_n^*$ al azar y verificamos si $a^{\frac{m}{2}} = 1$, para todos ellos. Si alguno de ellos no cumple esta propiedad, sabremos que $m/2$ no satisface (2.3) y el bucle acabará. En caso contrario, es decir, si el número de $a \in \mathbb{Z}_n^*$ con los que hemos probado es k, y para todos ellos se verifica $a^{\frac{m}{2}} = 1$, entonces podemos asegurar con una probabilidad $\geq 1 - \frac{1}{2^k}$ que $\frac{m}{2}$ satisface (2.3). En tal caso, sustituiremos m por $\frac{m}{2}$ y el bucle se reiniciará. El bucle continuará hasta que encontremos un número m que satisfaga (2.3) pero $m/2$ no lo satisfaga.

Como $n = pq$, con p y q primos distintos, del Teorema Chino de los Restos se tiene que

$$x^{m/2} \equiv 1 \mod n \quad \Leftrightarrow \quad \begin{cases} x^{m/2} \equiv 1 \mod p \\ \text{y} \\ x^{m/2} \equiv 1 \mod q. \end{cases}$$

Por tanto, si ya sabemos que $m/2$ no satisface la propiedad (2.3) existirá al menos un valor $a \in \mathbb{Z}_n^*$ tal que

$$a^{m/2} \not\equiv 1 \mod p \quad \text{ó} \quad a^{m/2} \not\equiv 1 \mod q.$$

Por simetría podemos suponer que $a^{m/2} \not\equiv 1 \mod p$ y consideramos dos casos.

Caso 1. Supongamos que $x^{m/2} \equiv 1 \mod q$, para todo $x \in Z_q^*$.

Entonces $\text{mcd}(a^{m/2} - 1, n) = p$ y habremos encontrado un divisor propio de n, con lo que conoceremos la factorización de n.

Caso 2. Supongamos que $x^{m/2} \not\equiv 1 \mod q$, para algún $x \in Z_q^*$.

En tal caso los conjuntos $A = \{x \in \mathbb{Z}_p^* : x^{m/2} \equiv 1 \mod p\}$ y $B = \{x \in \mathbb{Z}_q^* : x^{m/2} \equiv 1 \mod q\}$ son dos subgrupos propios de $\mathbb{Z}_p^*$ y $\mathbb{Z}_q^*$. Como p y q son primos $\mathbb{Z}_p^*$ y $\mathbb{Z}_q^*$ son cíclicos de orden $p-1$ y $q-1$ respectivamente. Como además m satisface la propiedad (2.3), $x^2 \in A$ e $y^2 \in B$ para todo $x \in \mathbb{Z}_p^*$ e $y \in \mathbb{Z}_q^*$. Eso implica que A y B tienen índice 2 en $\mathbb{Z}_p^*$ y $\mathbb{Z}_q^*$ respectivamente, con lo que $|A| = (p-1)/2$ y $|B| = (q-1)/2$. Por tanto, la probabilidad de que un elemento al azar de $\mathbb{Z}_n^*$ esté en $(A \setminus B) \cup (B \setminus A)$ es $1/2$. (Aquí se está utilizando el Teorema Chino de los Restos.). Una vez que tengamos un elemento $a \in (A \setminus B) \cup (B \setminus A)$ podemos hacer como en el caso anterior y tendremos que $\mathrm{mcd}(a^{m/2} - 1, n) = p$ ó q. ∎

Obsérvese que en el algoritmo implícito en la demostración del Teorema 2.8, no podemos en ningún momento utilizar los valores de p y q, y por tanto no podemos comprobar si $a^{m/2} \equiv 1 \mod p$ ó no. Lo que en realidad se hace, es que una vez que hemos encontrado un valor a tal que $a^{m/2} \not\equiv 1 \mod n$, entonces calculamos $\mathrm{mcd}(x^{m/2} - 1, n)$ para diversos valores de x. Lo que nos dice la demostración es que la probabilidad de que este valor no sea ni 1 ni n es $1/2$, con lo que después de varias pruebas encontraremos un divisor propio de n.

2.4 Logaritmo discreto

La fortaleza del protocolo de intercambio de claves de Diffie-Helman está basada en la dificultad del Problema del Logaritmo Discreto. Este problema se define de la siguiente forma.

Definición 2.9 *Sea S un semigrupo finito. El Problema del Logaritmo Discreto en el semigrupo S es el de resolver ecuaciones del tipo*

$$a^X = b \quad (x \in \mathbb{N})$$

donde a y b son dos elementos dados de S.

La dificultad del Problema del Logaritmo Discreto varía mucho según sea el semigrupo S. Por ejemplo, el Problema del Logaritmo Discreto en el grupo aditivo de $\mathbb{Z}_n$ es muy fácil de resolver pues consiste en resolver una ecuación de congruencias del tipo $aX \equiv b \mod n$ que equivale a resolver la ecuación diofántica $aX + nY = b$. (Ver Sección 1.8.) Sin embargo, el Problema del Logaritmo Discreto en los semigrupos multiplicativos de $\mathbb{Z}_n$ ó $\mathbb{F}_q$ o en sus grupo de unidades es considerablemente más difícil. Obsérvese que $\mathbb{F}_q^*$ es cíclico de orden $q-1$, por el Teorema 1.23. Por tanto, $\mathbb{F}_q^* \simeq (\mathbb{Z}_{q-1}, +)$ con lo que, desde un punto de vista matemático, tenderíamos a pensar que los Problemas del Logaritmo Discreto en $\mathbb{F}_q^*$ y $\mathbb{Z}_{q-1}$ deberían ser igual de difíciles. Esto sería cierto si conociéramos un isomorfismo $f : \mathbb{Z}_{q-1} \to \mathbb{F}_q^*$ explícito para el que se pudiera calcular tanto $f(x)$ como $f^{-1}(y)$ para todo $x \in \mathbb{Z}_{q-1}$ y todo $y \in \mathbb{F}_q^*$. Obsérvese que $a^X = b$ tiene solución si y sólo si b está en el subsemigrupo cíclico generado por a. Por tanto, si a es un elemento de orden finito de un grupo, en la práctica se puede suponer que S es un grupo cíclico y por tanto isomorfo a $(\mathbb{Z}_n, +)$. De nuevo la dificultad no está en la estructura del grupo, sino en reconocer los elementos como potencias de enteros. En resumen, la dificultad del Problema del Logaritmo Discreto en un grupo no depende tanto de su estructura algebraica (pues podemos suponer que es un grupo cíclico) sino de la forma de presentar sus elementos.

Se ha conjeturado que el Problema del Logaritmo Discreto es un problema $\mathbb{NP}$-completo, pero esta conjetura todavía está por demostrar. Los algoritmos que se conocen para resolver el Problema del Logaritmo Discreto, incluso para grupos sencillos como $\mathbb{Z}_n^*$, son muy lentos. Por tanto, hoy en día la función $E_a : \mathbb{N} \to S$, dada por $E_a(n) = a^n$ se considera una función de dirección única, si el semigrupo S y a se eligen apropiadamente.

De forma naïf, se puede diseñar un criptosistema utilizando la dificultad del Problema del Logaritmo Discreto a partir de un grupo finito S de orden n tomando S como conjunto de mensajes en claro y cifrados, $\mathbb{Z}_n^*$ como conjunto de claves públicas y privadas y tomando como aplicaciones de cifrado y descifrado

$$\begin{array}{rccc} c_x = d_x = E_x : & S & \to & S \\ & g & \mapsto & g^x. \end{array}$$

La función que transforma la clave pública en la clave privada es

$$\begin{array}{rccc} f : & K' = \mathbb{Z}_n^* & \to & K = \mathbb{Z}_n^* \\ & k & \mapsto & (k^{-1} \mod n) \end{array}$$

Obsérvese que del Teorema de Lagrange, $x^n = 1$ para todo $x \in S$. Como además $mf(m) \equiv 1 \mod n$ se tiene que $mf(m) = tn + 1$ para algún entero, con lo cual

$$d(c(m,k), f(k)) = (m^k)^{f(k)} = m^{kf(k)} = m^{nt+1} = (m^n)^t m = m.$$

Sin embargo hay que tener cuidado por que esto no sirve como criptosistema de clave pública tal como está ya que la función f es tan fácil de calcular como de invertir utilizando el Algoritmo de Euclides (Sección 1.8). Una alternativa es el siguiente criptosistema:

Criptosistema de Massey-Omura

Todos los usuarios del sistema se ponen de acuerdo en un grupo finito G de orden n. Éste es el único elemento público de este sistema y es el mismo para todo el mundo. Cada usuario elige como clave un elemento de $\mathbb{Z}_n^*$ que mantiene siempre secreta. Supongamos que Alicia quiere mandar el mensaje m a Blas y que sus respectivas claves son k_A y k_B, de forma que k'_A y k'_B son los inversos de k_A y k_B módulo n. Entonces:

- Alicia envía $a = m^{k_A}$ a Blas.
- Blas le devuelve $b = a^{k_B}$.
- Alicia envía $c = b^{k'_A}$.
- Finalmente Blas descifra

$$c^{k'_B} = (b^{k'_A})^{k'_B} = ((a^{k_B})^{k'_A})^{k'_B} = (((m^{k_A})^{k_B})^{k'_A})^{k'_B} = m^{k_A k_B k'_A k_B} = m.$$

La ventaja de este criptosistema es que no existe ningún tipo de clave. Claro que eso se puede convertir en una desventaja. (Ver Tarea 1.)

Criptosistema de El Gammal

Otra forma de utilizar el logaritmo discreto en un criptosistema de clave pública es el *Criptosistema de El Gammal*. En realidad este método de cifrado se sale de nuestra definición de criptosistema por que en la aplicación de cifrado se hará una elección aleatoria. Esto es una práctica frecuente en diversos criptosistemas que formalizamos en la siguiente definición.

Definición 2.10 *Un criptosistema de clave pública con elección aleatoria es como un criptosistema de clave pública, con una función de dirección única $P : K' \to K$, que asocia claves privadas con claves públicas, funciones de descifrado $d_k : C_k \to M_k$ $(k \in K)$ y funciones de cifrado. La diferencia con los criptosistemas de clave pública estándar está en que las funciones de cifrado son aplicaciones*

$$c_k : M_k \times A_k \to C_k$$

donde A_k es un conjunto no vacío y la condición $d_{k'}c_k(m) = m$ se sustituye por la siguiente

$$d_{k'}c_k(m, a) = m,$$

para todo $k' \in K', k = P(k'), m \in M_k$ y $a \in A_k$.

Cuando un usuario desea cifrar un mensaje en claro $m \in M_k$ con un criptosistema de clave pública con elección aleatoria, elige un elemento de $a \in A_k$ y cifra calculando $c_k(m, a)$.

Para el Criptosistema de El Gammal se elige un grupo finito G y un elemento $g \in G$. El elemento g es público y común para todo el mundo. Normalmente g se elige de orden grande. Por ejemplo, si tomamos $G = \mathbb{F}_q^*$, el grupo multiplicativo del cuerpo con q elementos, lo ideal sería tomar como g un generador de $\mathbb{F}_q^*$. Los conjuntos de claves privadas y públicas son $K' = \mathbb{Z}_n^*$ y $K = \langle g \rangle$, donde n es el orden de g. En caso de no conocer el orden de g también se puede tomar el orden de G y, si tampoco conociéramos el orden de este, también podríamos tomar $K' = \mathbb{Z}$. Para todas las claves, los conjuntos de mensajes en claro y cifrados son $M = G$ y $C = G \times G$, respectivamente. La función que asocia las claves privadas con las públicas viene dada por

$$\begin{array}{rccc} P: & K' & \to & K \\ & k' & \mapsto & k = g^{k'} \end{array}$$

El conjunto utilizado para la elección aleatoria es el conjunto de los números enteros y la función de cifrado con la clave pública k es

$$\begin{array}{rccc} c_k: & M \times \mathbb{Z} & \to & C \\ & (m, x) & \mapsto & (g^x, mk^x) \end{array}$$

Finalmente, la función de descifrado con la clave privada k' es

$$\begin{array}{rccc} d_{k'}: & C & \to & M \\ & (u, v) & \mapsto & vu^{-k'} \end{array}$$

Observemos que si $k = P(k') = g^{k'}$ entonces se verifica

$$d_{k'}(c_k(m, x)) = d_{k'}((g^x, m(g^{k'})^x)) = mg^{k'x}(g^x)^{-k'} = m.$$

DSA

En 1991 el National Institute of Standards and Technology (NIST) del gobierno americano estableció el siguiente estándar para firma digital con garantía de integridad, conocido como DSA (Digital Signature Algorithm) basado en el Problema del Logaritmo Discreto.

Recuérdese que si G es un grupo cíclico de orden n, entonces para cada divisor d de n, G tiene un único subgrupo de orden d.

Elección de una clave pública para firma digital.

- Se elige un primo q de unos 160 bits.
- Se elige un segundo primo $p \equiv 1 \mod q$ de al menos 500 bits. En realidad, por razones de implementación, se recomienda que el número de bits de p sea múltiplo de 64 y entre 512 y 1024.

  El Teorema de Dirichlet sobre primos en progresión aritmética garantiza la existencia de infinitos primos p congruentes con 1 módulo q, pues este teorema afirma que toda sucesión aritmética infinita del tipo $(a + nb, n \in \mathbb{N})$, con a y b enteros coprimos contiene infinitos primos.
- Se elige un generador g del único subgrupo de $\mathbb{F}_p^*$ de orden q. La forma de obtener este generador es calculando
$$(x^{\frac{p-1}{q}} \mod p)$$
  para un $x \in \mathbb{F}_p^*$ elegido al azar. Si este número no es 1, entonces $g = x^{\frac{p-1}{q}}$ es un generador apropiado.
- Se elige un entero al azar $1 \leq k' \leq q-1$ que, junto con p y q forma la clave privada del usuario. La clave publica es $(p, q, k = g^{k'})$.

¿Cómo se firma un mensaje?.

- Se aplica una función resumen estandarizada al mensaje y se obtiene un número resumen $0 < H < q$.
- Se elige un número al azar $1 < a < q$ y se calculan
$$\begin{aligned} x &= (g^a \mod p) \\ r &= (x \mod q) \\ s &= (a^{-1}(H + rk') \mod q). \end{aligned}$$
- La firma del mensaje es (r, s).

Verificación de la firma.
Se calculan:

- El resumen H del mensaje;

- $u_1 = (s^{-1}H \mod q)$ y $u_2 = (s^{-1}r \mod q)$ y
- $x_1 = (g^{u_1}k^{u_2} \mod p)$.

Se acepta la firma como correcta si $x_1 \equiv r \mod q$. Esto está justificado por que, como g tiene orden q y $p \equiv 1 \mod q$, se verifica .

$$x_1 \equiv g^{u_1}k^{u_2} \equiv g^{s^{-1}H+k's^{-1}r} \equiv g^{s^{-1}(H+k'r)} \equiv g^a \equiv x \mod p$$

Como además, $0 \leq x_1, x < p$, se tiene que $x_1 = x$ y por tanto $x_1 \equiv r \mod q$.

Todos los pasos en los cálculos anteriores se pueden hacer en tiempo polinomial excepto la búsqueda del generador del único subgrupo de $\mathbb{F}_p^*$ de orden q que se puede hacer en tiempo polinomial probabilístico. Sea G dicho grupo. Entonces

$$|\mathbb{F}_p^*/G| = \frac{p-1}{q}$$

con lo que $a^{\frac{p-1}{q}} \in G$ para todo $a \in \mathbb{F}_p^*$. Además, como $|G| = q$ es primo, el único no generador de G es 1. Si elegimos $a \in \mathbb{F}_p^*$ al azar la probabilidad de que $a^{\frac{p-1}{q}}$ no sea generador de G es $\frac{1}{q}$.

2.5 Criptosistema de la mochila

Definición 2.11 *El Problema de la Mochila*[3] *consiste en encontrar una solución de la ecuación*

$$x_1v_1 + x_2v_2 + \ldots + x_hv_h = V$$

con $x_i = 0, 1$*, donde* $v_1, v_2, \ldots, v_h$ *y* V *son números naturales.*

El nombre del Problema de la Mochila proviene de interpretarlo como el problema de rellenar exactamente una mochila de un volumen dado V con una lista de objetos de volúmenes $v_1, \ldots, v_h$. Se sabe que el Problema de la Mochila es $\mathbb{NP}$-completo. Sin embargo, un subproblema del Problema de la Mochila muy fácil de resolver es el Problema de la Mochila para Sucesiones Supercrecientes. Una sucesión $v_1, \ldots, v_h$ de números naturales se dice *supercreciente* si para cada $i \leq h$ se verifica $\sum_{j=1}^{i-1} v_j < v_i$. Entendemos que la suma vacía es 0, con lo que si $(v_1, \ldots, v_h)$ es supercreciente entonces $0 < v_1$. El siguiente algoritmo resuelve el Problema de la Mochila para Sucesiones Supercrecientes:

[3] "Knapsack Problem" en la literatura en inglés

Algoritmo 7. Solución del Problema de la Mochila para Sucesiones Supercrecientes

ENTRADA: $v_1, \ldots, v_h, V \in \mathbb{N}$ con $v_1, \ldots, v_h$ una sucesión supercreciente.
$c := 0$
Para $i = h, \ldots, 1$,
si $c + v_i \leq V$, entonces $x_i := 1$ y $c := c + v_i$;
en caso contrario $x_i := 0$.
Si $V = x_1 v_1 + \ldots + x_h v_h$, entonces
SALIDA: $(x_1, \ldots, x_h)$,
En caso contrario
SALIDA: El problema no tiene solución.

El siguiente criptosistema conocido como *Criptosistema de la Mochila* fue propuesto por Helman y Merkle en 1978. El conjunto K de claves públicas es el de sucesiones finitas de números naturales. Si $k = (w_1, \ldots, w_h) \in K$ entonces el conjunto M_k de mensajes en claros para la clave k es $M_k = \mathbb{Z}_2^h$, el conjunto C_k de mensajes cifrados es el conjunto $\mathbb{N}$ de los números naturales y la aplicación de cifrado es

$$\begin{array}{rccc} c_k: & M_k = \mathbb{Z}_2^h & \to & C_k = \mathbb{N} \\ & m = (m_1, \ldots, m_h) & \to & m_1 w_1 + \ldots + m_h w_h \end{array}$$

La aplicación de descifrado es

$$\begin{array}{rccl} d_k: & C_k = \mathbb{N} & \to & M_k = \mathbb{Z}_2^h \\ & c & \to & \text{Solución al problema de la mochila para la sucesión } w_1, \ldots, w_h, c \end{array}$$

Obsérvese que d_k no es precisamente una aplicación pues sólo está definida para los elementos de la imagen de c_k y algunos de los elementos de esta imagen pueden tener varias imágenes por d_k. Además, tal como está planteado este no es un criptosistema útil ya que no está claro como calcular d aunque se conozca la clave. Claro que esto es lo que queremos en un criptosistema de clave pública y lo que hacemos es utilizar que el Problema de la Mochila para Sucesiones Supercrecientes tiene una solución fácil para convertir este criptosistema en un criptosistema de clave pública tomando

$$K'_k = \{(v_1, \ldots, v_h, n, a, b) \in \mathbb{N}^{k+3} : v_1, \ldots, v_h, n \text{ supercreciente } 0 < a, b < n \text{ y } ab \equiv 1 \mod n\}$$

como conjunto de claves privadas. Entonces la aplicación que asocia claves privadas con claves públicas es

$$\begin{array}{rccc} P: & K'_k & \to & K_k \\ & (v_1, \ldots, v_h, n, a, b) & \mapsto & ((av_1, \ldots, av_h) \mod n) \end{array}$$

es decir todas las componentes av_i se reducen módulo n y la aplicación de descifrado con la clave privada $k' = (v_1, \ldots, v_h, n, a, b)$ es

$$\begin{array}{rccl} d_{k'}: & C_{k'} = \mathbb{N} & \to & M_{k'} = \mathbb{Z}_2^h \\ & c & \mapsto & \text{Solución del Problema de la Mochila para } v_1, \ldots, v_h, bc \end{array}$$

Vamos a ver que efectivamente se verifica $d_{k'}(c_k(m)) = m$. Si $k' = (v_1, \ldots, v_h, n, a, b)$, entonces

$$k = P(k') = (w_1, \ldots, w_h) = ((av_1, \ldots, av_h) \mod n),$$

con lo que

$$c = c_k(m_1, \ldots, m_h) = m_1w_1 + \ldots m_hw_h = (a(m_1v_1 + \ldots + m_hv_h) \mod n).$$

Por tanto, como $ab \equiv 1 \mod n$, tenemos

$$\begin{array}{rcl} d_{k'}(c) & = & \text{Solución del Problema de la Mochila para } v_1, \ldots, v_h, (ba(m_1v_1 + \ldots + m_hv_h) \mod n) \\ & = & \text{Solución del Problema de la Mochila para } v_1, \ldots, v_h, m_1v_1 + \ldots + m_hv_h \\ & = & (m_1, \ldots, m_h). \end{array}$$

Obsérvese que la unicidad en la última igualdad está garantizada por ser $v_1, \ldots, v_h$ una sucesión supercreciente.

Sobre el criptoanálisis del criptosistema de la mochila

La ventaja del Criptosistema de la Mochila sobre los otros criptosistemas de clave pública que hemos visto (RSA y los basados en el Problema del Logaritmo Discreto) es triple. Por un lado los algoritmos de implementación del Criptosistema de la Mochila son mucho más rápidos y, por otro lado, encontrar claves es mucho más fácil para el Criptosistema de la Mochila que para los otros dos, ya que para RSA necesitamos buscar números primos muy grandes y para los criptosistesmas basados en el Problema del Logaritmo Discreto necesitamos utilizar semigrupos finitos grandes. Finalmente, mientras que el problema de ruptura del Criptosistema de la Mochila es teóricamente el del Problema de la Mochila que se sabe que es $\mathbb{NP}$-completo, el problema de ruptura de los otros dos, aunque supuestamente difícil, todavía no está demostrado que sean problemas $\mathbb{NP}$-completos.

Estas ventajas despertaron muchas expectativas sobre el uso masivo del Criptosistema de la Mochila. Sin embargo en 1982, Shamir (la "S" de RSA) observó que el problema de ruptura del Criptosistema de la Mochila no es exactamente el del Problema de la Mochila sino un subproblema suyo. Concretamente para romper el Criptosistema de la Mochila es necesario resolver el Problema de la Mochila para permutaciones lineales módulo n de sucesiones supercrecientes. Shamir proporcionó un algoritmo de tiempo polinomial que resuelve este subproblema del Problema de la Mochila[4]. Otras versiones del Criptosistema de la Mochila fueron introducidas para intentar salvar su seguridad. Por ejemplo una de ellas está basada en hacer dos transformaciones lineales $x \mapsto (a_1x \mod n_1)$ y $x \mapsto (a_2x \mod n_2)$ con $n_1 \neq n_2$ y $\text{mcd}(a_i, n_i) = 1$. Sin embargo estas versiones también cayeron rápidamente[5].

[4] A. Shamir, A polynomial time algorithm for breaking the basic Merkle-Helman cryptosystem, Proceedings of the 23rd Annual Sypmposium on the Foundations of Computer Science (1982), 145–152

[5] ver E.F. Brickell, Breaking iterated knapsack, Advances in Cryptology, Crypto '84 (1985) Springer-Verlag, 342-358. y A.M. Odlyzko, The rise and fall of knapsack cryptosystem, Cryptology and Computational Number Theory, Proc. Symp. Appl. Math. 42 (1990) 75–88

2.6 Tareas

(1) ¿Sería posible utilizar el criptosistema de Massey-Omura para firma digital? ¿Se te ocurre alguna forma de suplantación usando Diffie-Helman o Massey-Omura?

(2) Implementar los siguiente criptosistemas en un programa de ordenador:

(a) RSA.

(b) Masey-Omura.

(c) El Gammal.

Los criptosistemas tienen que ser capaces de transformar un texto comprensible escrito en un alfabeto que se haya prescrito en un mensaje cifrado con la clave pública dada y de descifrarlo con la clave privada.

(3) Implementar firma digital en un programa con algún criptosistema de clave pública. Implementar DSA.

(4) Demostrar que el problema de resolver ecuaciones cuadráticas en $\mathbb{Z}_n$ se reduce en tiempo polinomial al subproblema de calcular raíces cuadradas en $\mathbb{Z}_n$. Es decir mostrar un algoritmo que resuelva ecuaciones cuadráticas en $\mathbb{Z}_n$ usando un posible algoritmo que calcule raíces cuadradas en $\mathbb{Z}_n$ (eso se llama un oráculo) de forma que sin contar el tiempo de cálculo del oráculo el tiempo del algoritmo sea polinomial.

(5) Sean p y q dos primos distintos y sea $n = pq$. Demostrar que el problema de calcular raíces cuadradas en $\mathbb{Z}_n$ se reduce en tiempo polinomial al de resolver raíces cuadradas en $\mathbb{Z}_p$ y en $\mathbb{Z}_q$.

(6) El criptosistema RSA(h) con $h \in \mathbb{N}$, es la siguiente variación de RSA. Sea $n = pq$ dado con p y q primos distintos que no se hacen públicos. La clave de cifrado (pública) es una h-tupla $(x_1, \ldots, x_h) \in \mathbb{Z}_{\phi(n)}$ y la de descifrado otra h-tupla $(y_1, \ldots, y_h) \in \mathbb{Z}_{\phi(n)}$ con $x_1y_1 + \ldots + x_hy_h \equiv 1 \mod \phi(n)$. Los conjuntos de mensajes en claro y cifrado son $M = \mathbb{Z}_n$ y $C = \mathbb{Z}_n^h$ y las aplicaciones de cifrado y descifrado con la claves pública $k = (x_1, \ldots, x_h)$ y privada $k' = (y_1, \ldots, y_h)$ son

$$\begin{aligned} c_k(m) &= (m^{x_1}, \ldots, m^{x_h}) \\ d_{k'}(x_1, \ldots, x_h) &= x_1^{y_1} \cdots x_h^{y_h}, \end{aligned}$$

donde todos los cálculos se hacen módulo n.

(a) Demostrar que este criptosistema es igual de seguro que RSA, es decir que el problema de romper uno de los criptosistemas se reduce en tiempo polinomial al de romper el otro.

(b) Mostrar cómo se puede romper el criptosistema con una clave pública que satisfaga $\gcd(x_1, \ldots, x_h) = 1$.

(7) Demostrar que 2 es un generador de $\mathbb{Z}_{3^k}^*$. Encontrar un algoritmo que resuelva la siguiente ecuación para todo $a \in \mathbb{Z}_{3^k}$

$$2^X \equiv a \mod 3^k$$

en tiempo polinomial. Es decir resuelve el problema del logaritmo discreto con base 2 en el grupo $\mathbb{Z}_{3^k}$. Indicación: Obtener la solución en base 3.

(8) En esta tarea vamos a ver una forma de acelerar el tiempo de cálculo de $x^a \mod n$ en el contexto del cifrado o descifrado de RSA, es decir supondremos que $n = pq$ con p y q primos distintos.

(a) Demostrar que la salida m del siguiente algoritmo satisface $m \equiv x^a \mod pq$.

Algoritmo de Aceleración de RSA:

ENTRADA: $a \in \mathbb{N}$ y dos primos distintos p y q.

$a_p := a \mod (p-1)$, $a_q := a \mod (q-1)$

$m_p := x^{a_p} \mod p$, $m_q := x^{a_q} \mod q$

(u_p, v_p) = Algoritmo de Euclides Extendido(p, q), es decir $u_p p + v_p q = 1$.

SALIDA: $m := m_q u_p p + m_p u_q q \mod pq$.

(b) Demostrar que el tiempo de cálculo de este algoritmo es $O(\max(p,q)^3)$ y compararlo con el tiempo de cálculo de $x^a \mod n$, si en ambos casos se han realizado las exponenciaciones modulares doblando cuadrados.

(9) En esta tarea vamos a ver una forma de atacar RSA que está basado en el hecho de que las aplicaciones de cifrado y descifrado $x \mapsto x^a \mod n$ son homomorfismos multiplicativos. Supongamos que $c = m^e \mod n$ es un mensaje cifrado con la clave (n, e) de RSA. Queremos descubrir cuál es el mensaje en claro m y supongamos que su longitud binaria es $l = l(m)$ que se puede escribir como $m = uv$, con u y v números de longitud binaria menor o igual que $\lceil \frac{l}{2} \rceil$, aunque nosotros no conocemos ni m ni u ni v.

(a) Demostrar que si $\mathrm{mcd}(c, n)$ no es ni 1 ni n entonces podemos encontrar m sin dificultad. Por tanto en el resto de la tarea se supone que $\mathrm{mcd}(c, n) = 1$.

(b) Supongamos que tenemos una tabla formada por las parejas $\{(r, r^e \mod n) : l(r) \leq \lceil \frac{l}{2} \rceil\}$. Mostrar cómo utilizar esta tabla para encontrar dos números r y s con $l(r), l(s) \leq \lceil \frac{l}{2} \rceil$ y $c \equiv (rs)^e \mod n$ y probar que en tal caso $m \equiv rs \mod n$.

(10) Demostrar que si la entrada del siguiente algoritmo es (g, a) y la salida es un entero m entonces $g^m = a$. Demostrar también que si la salida es el mensaje que viene al final entonces lo que dice el mensaje es correcto.

Algoritmo de Pasos de Niño-Pasos de Gigante

ENTRADA: (g, a) una pareja formada por dos elementos de un grupo y n un entero positivo.

$s := \lceil \sqrt{n} \rceil$.

$L :=$ Lista ordenada $\{ag^j : j = 0, \ldots, s-1\}$.

$c := g^s$ (que se puede calcular multiplicando el último elemento de L por $a^{-1}g$),

$i := 1$; $d := c$;

Mientras que $i \leq s$

Si d aparece en L en la posición j-ésima entonces

SALIDA: $is - j + 1$.

$d := dc$; $i := i + 1$;

Si el algoritmo no se para en el bucle anterior entonces:

SALIDA: "a no está en el grupo generado por g ó n es mayor que el orden de g".

Capítulo 3

Primalidad

En los capítulos anteriores se ha visto la necesidad de disponer de métodos para proporcionar números primos y, por tanto, de disponer de algún algoritmo que decida si un número es primo o no. En este capítulo vamos a ver revisar algunos de estos algoritmos.

3.1 Criba de Eratóstenes

El algoritmo más obvio para decidir si un número n es primo es simplemente aplicar la definición y recorrer los números menores que n y mayores que 1 buscando algún divisor entre ellos. Si lo encontramos decidiremos que el número es compuesto y en caso contrario en número habrá de ser primo. En realidad si n es compuesto entonces su menor divisor distinto de 1 será primo y menor o igual que $\sqrt{n}$, con lo que basta buscar divisores menores o iguales que $\sqrt{n}$ y si no los hay concluir que n será primo. Esto reduce considerablemente la cantidad de números a considerar pero todavía tenemos que hacer unas $\sqrt{n}$ divisiones lo que da un tiempo de cálculo $O(e^{\frac{k}{2}})$ para un número de longitud k. Es decir, el tiempo de cálculo de este algoritmo es exponencial.

Una pequeña modificación de este método proporciona también la lista de números primos menores o iguales que n. Se trata de la *Criba de Eratóstenes*. Para ello se comienza escribiendo todos los números entre 2 y n. Entonces se repite el siguiente procedimiento hasta que el primer número sin marcar ni tachar sea mayor que la raíz cuadrada de n: Se busca el primer número d sin marcar ni tachar que al principio será $d = 2$ pero después tomará otros valores. Se marca dicho número y se tachan los números mayores o iguales que d^2 que sean múltiplos de d. Es decir, se marca d, se tacha d^2, y a partir de aquí se tachan los números de d en d. Cuando el proceso acabe los números que quedan sin tachar ni marcar son los primos menores o iguales que n.

Por ejemplo, si queremos construir la lista de primos menores que 100 comenzamos escri-

biendo la lista de los números de 2 a 100.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Se marca el 2 y se tachan los demás números pares:

	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	25	~~26~~	27	~~28~~	29	~~30~~
31	~~32~~	33	~~34~~	35	~~36~~	37	~~38~~	39	~~40~~
41	~~42~~	43	~~44~~	45	~~46~~	47	~~48~~	49	~~50~~
51	~~52~~	53	~~54~~	55	~~56~~	57	~~58~~	59	~~60~~
61	~~62~~	63	~~64~~	65	~~66~~	67	~~68~~	69	~~70~~
71	~~72~~	73	~~74~~	75	~~76~~	77	~~78~~	79	~~80~~
81	~~82~~	83	~~84~~	85	~~86~~	87	~~88~~	89	~~90~~
91	~~92~~	93	~~94~~	95	~~96~~	97	~~98~~	99	~~100~~

El primer número sin marcar ni tachar es el 3. Por tanto se marca el tres y se tachan de tres en tres comenzando por 9.

	2	**3**	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	25	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	35	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	49	~~50~~
~~51~~	~~52~~	53	~~54~~	55	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	65	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	77	~~78~~	79	~~80~~
81	~~82~~	83	~~84~~	85	~~86~~	~~87~~	~~88~~	89	~~90~~
91	~~92~~	~~93~~	~~94~~	95	~~96~~	97	~~98~~	~~99~~	~~100~~

Después de dos pasos más se obtiene la siguiente tabla.

	2	**3**	~~4~~	**5**	~~6~~	**7**	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	~~49~~	~~50~~
~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	~~77~~	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~

Como el primer número sin marcar ni tachar es $11 > \sqrt{100}$, el algoritmo ha acabado. En consecuencia la lista de los primos menores o iguales que 100 es: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.

El siguiente código implementa la criba de Eratóstenes para GAP.

```
CEratostenes := function(n)
local lista, p, primos, multiplos;
lista := [2..n];
p := 2;
primos := [2];
while p^2 <= n do
  multiplos := List([1..QuoInt(n,p)],x->p*x);
  lista := Difference(lista,multiplos);
  p := lista[1];
  Add(primos,p);
od;
return Concatenation(primos,lista);
end;
```

Utilizando esta función podemos calcular todos los primos menores o iguales que un número dado. Por ejemplo, así podemos calcular todos los primos menores que 10 millones en menos de 10 segundos.

```
gap> Primos:=CEratostenes(10^7);;
time;
gap> time;
8580
gap> Size(Primos);
664580
gap> Primos[1000];
7907
gap> Primos[5*10^5];
7368743
```

y descubrir que hay 664.580 primos menores que 10 millones, que el que ocupa el lugar 1000 es 7.907 y el que ocupa el lugar medio millón es el 7.368.743. El Teorema del Número Primo asegura que si $\pi(n)$ denota el número de primos menores o iguales que n entonces $\lim_{n\to\infty} \frac{\pi(n)}{n/\log(n)} = 1$, es decir que $\pi(n)$ se aproxima a $\frac{n}{\log(n)}$, que hemos comprobado con el cálculo anterior pues $\frac{10^7}{\log(10^7)} \approx 620.421$. Por desgracia no podemos ir mucho más allá aunque sólo sea porque GAP no permite construir listas de longitud mayor que $2^{28} - 1 = 26.8435.455$.

```
gap> CEratostenes(10^9);;
Error, Range: <last> must be an integer less than 2^28
  (not a integer (>= 2^28)) in lista := [ 2 .. n ]; called from
<function "CEratostenes">( <arguments> )
 called from read-eval loop at line 43 of *stdin*
you can replace <last> via 'return <last>;'
```

En cualquier caso, podemos utilizar la lista de primos obtenida para decidir si otros números lo son intentando ver si tienen algún divisor en la lista como hace el siguiente programa.

```
EsPrimoPeque := function(n)
local i,p;
i:= 1;
p:=2;
while i<=664580 and p^2<n do
  p:=Primos[i];
  if RemInt(n,p)=0 then
    Print("Menor Divisor Primo = ",p,"\n");
    return false;
  fi;
  i:=i+1;
od;
if 10^14 >= n then
  return true;
else
  Print("No tiene divisores menores que 10^14,\n");
  return true;
fi;
end;
```

Si el número no es mayor que 10^{14} y no es divisible por ninguno de la lista `Primos` obtenida entonces será primo. Sin embargo si es mayor que 10^{14} sólo podremos verificar si es divisible por esos números. Si lo es, será compuesto pero si no lo es no podremos asegurar que es primo.

```
gap> EsPrimoPeque(109289080983098101);
No tiene divisores menores que 10^14,
true
```

```
gap> EsPrimoPeque(190890980989);
true
gap> EsPrimoPeque(19009998098098089);
Menor Divisor Primo = 23
false
```

Con esto estamos en condiciones de decidir si un número con a lo sumo 14 cifras decimales es o no primo. Pero también estamos en condiciones de factorizarlo modificando ligeramente el algoritmo anterior con lo cual estos primos son demasiado pequeños para ser útiles en términos de RSA.

3.2 Pseudoprimos. Números de Carmichael

Definición 3.1 *Un* test de primalidad *es un algoritmo que decide si un número es primo. Es decir en un test de primalidad, la entrada es un entero y la salida es "VERDADERO" o "SÍ" en el caso en el que la entrada sea un número primo y "FALSO" o "NO" en caso contrario.*

Un test de primalidad probabilístico *es un test probabilístico que tiene como entrada un entero positivo y que hace una elección al azar de algún dato de forma que si la salida es "VERDADERO", entonces la probabilidad de que el número sea primo es* $\geq 1/2$.

Si n es un número entero no nulo, entonces a las unidades de $\mathbb{Z}_n^*$ los vamos a llamar *bases módulo* n. Por extensión, también llamaremos bases módulo n a los enteros coprimos con n.

Definición 3.2 *Sea* $0 \neq n \in \mathbb{Z}$ *y sea* b *una base módulo* n. *Decimos que* n *es* pseudoprimo *en la base* b *si*

$$b^{n-1} \equiv 1 \mod n.$$

Por el Teorema Pequeño de Fermat, si p es primo entonces p es pseudoprimo en todas las bases módulo p. Obsérvese que decidir si un número es pseudoprimo en una base es una tarea rápida de ejecutar utilizando el Algoritmo 5. Además, si n no es pseudoprimo en alguna base, y por tanto no es primo, entonces podemos encontrar esa base de forma rápida. La razón es la siguiente proposición que implica que el número de bases en las que n es pseudoprimo divide al número total de bases.

Proposición 3.3 *El conjunto de bases módulo* n *en las que* n *es pseudoprimo es un subgrupo de* $\mathbb{Z}_n^*$ *y en consecuencia si* n *no es pseudoprimo en todas las bases entonces no es pseudoprimo en al menos la mitad de las bases.*

Demostración. El conjunto de las bases en las que n es pseudoprimo es el núcleo del siguiente homomorfismo de grupos:

$$\begin{array}{rcl} \mathbb{Z}_n^* & \mapsto & \mathbb{Z}_n^* \\ b & \mapsto & b^{n-1} \end{array}$$

∎

A la vista de la Proposición 3.3, se puede diseñar un test de primalidad probabilístico de forma que el dato aleatoria elegido sea una base módulo el número que estamos tratando de decidir si es primo. El test tiene salida positiva si la entrada es pseudoprimo en la base elegida al azar. Si un número no es pseudoprimo en alguna base y pasa el test para k bases entonces será primo con una probabilidad $\geq 1 - \frac{1}{2^k}$. Sin embargo, para que esto se pueda considerar un test de primalidad es necesario que no haya números compuestos que sean pseudoprimos en todas las bases. Dichos números tienen nombre:

Definición 3.4 *Un* número de Carmichael *es un número compuesto que es pseudoprimo en todas la bases.*

Por desgracia, sí que hay números de Carmichael que están caracterizados por el siguiente teorema. Recordemos que un *entero* se dice *libre de cuadrados* si no es múltiplo de un cuadrado de un número mayor que 1.

Teorema 3.5 *Un número compuesto es de Carmichael si y sólo si es libre de cuadrados y para todo divisor primo p de n se verifica $p-1 \mid n-1$.*

Antes de demostrar el Teorema 3.5, necesitamos demostrar dos lemas. Introducimos la siguiente notación para n y m dos enteros con $\mathrm{mcd}(n,m) = 1$ y todo primo p:

$$\begin{array}{rcl} o_n(m) & = & \text{Orden multiplicativo de } m \text{ módulo } n = \min\{k > 0 : m^k \equiv 1 \mod n\} \\ v_p(n) & = & \text{Multiplicidad de } p \text{ en } n = \max\{k \geq 0 : p^k \mid n\}. \end{array}$$

Recordemos además que φ denota la función de Euler, es decir

$$\varphi(n) = |\mathbb{Z}_n^*| = \prod_{p|n} (p-1)p^{v_p(n)-1},$$

donde $\prod_{p|n}$ significa que p recorre los primos que dividen a n.

El siguiente lema recoge algunas propiedades obvias:

Lema 3.6 *Sean n, m y a enteros positivos y sea p un número primo. Entonces*

(1) $v_p(nm) = v_p(n) + v_p(m)$.

(2) $v_p(n+m) \geq \min\{v_p(n), v_p(m)\}$.

(3) Si $v_p(n) \neq v_p(m)$ *entonces* $v_p(n+m) = \min\{v_p(n), v_p(m)\}$.

(4) Si $a < p^n$, *entonces* $v_p(p^n - a) = v_p(i)$.

(5) Si $\gcd(a,n) = 1$ *entonces*

- *(a)* $a^m \equiv 1 \mod n$ *si y sólo si* $o_n(a)$ *divide a* m.
- *(b) Si* $n \mid m$ *entonces* $o_n(a) \mid o_m(a)$.
- *(c)* $o_n(a) \mid \varphi(n)$ *y* $o_n(a) = \varphi(n)$ *si y solo si* a *es un generador de* $\mathbb{Z}_n^*$.

Lema 3.7 *Sea p un número primo y sea a un entero que genera $\mathbb{Z}_{p^e}^*$.*

(1) Si $e = 1$ entonces a ó $a + p$ generan $\mathbb{Z}_{p^2}^*$.

(2) Si $e \geq 2$ y $p \neq 2$ entonces a genera $\mathbb{Z}_{p^{e+1}}^*$.

Demostración. Supongamos que a genera $\mathbb{Z}_{p^e}^*$. En particular $\gcd(a, p) = 1$. Por el apartado (5b) del Lema 3.6 se tiene que $o_{p^e}(a)$ divide a $o_{p^{e+1}}(a)$. Como además

$$p^{e-1}(p-1) = \varphi(p^e) = o_{p^e}(a) \quad \text{and} \quad o_{p^{e+1}}(a) \mid \varphi(p^{e+1}) = p^e(p-1),$$

tenemos que

$$o_{p^{e+1}}(a) = \begin{cases} p^{e-1}(p-1), & \text{si } a^{p^{e-1}(p-1)} \equiv 1 \mod p^{e+1}; \\ p^e(p-1) = \varphi(p^{e+1}), & \text{en otro caso.} \end{cases}$$

Luego a es generador de $\mathbb{Z}_{p^{e+1}}^*$ si y solo si $a^{p^{e-1}(p-1)} \not\equiv 1 \mod p^{e+1}$.

Supongamos que $p \neq 2$, $e = 1$ y ni a ni $a + p$ son generadores de $\mathbb{Z}_{p^2}$. Entonces $a^{p-1} \equiv (a+p)^{p-1} \equiv 1 \mod p^2$. Luego

$$1 \equiv (a+p)^{p-1} \equiv a^{p-1} + (p-1)a^{p-2}p \equiv 1 - a^{p-2}p \mod p^2,$$

y por tanto $p \mid a$, una contradicción. Esto demuestra (1) para el caso en que $p \neq 2$. El caso $p = 2$ es obvio.

Supongamos ahora que $e \geq 2$. Por el Teorema de Euler tenemos $a^{p^{e-2}(p-1)} = 1 + kp^{e-1}$ para un entero k. Pero $p \nmid k$, pues a genera $\mathbb{Z}_{p^e}$. Como $e \geq 2$, tanto $3(e-1)$ como $2e-1$ son mayores o iguales que $e+1$ y, por tanto,

$$\begin{aligned} a^{p^{e-1}(p-1)} &= (1 + kp^{e-1})^p \equiv 1 + pkp^{e-1} + \tbinom{p}{2} k^2 p^{2(e-1)} \\ &\equiv 1 + kp^e + \tfrac{p-1}{2} k^2 p^{2e-1} \equiv 1 + kp^e \not\equiv 1 \mod p^{e+1}. \end{aligned}$$

De la primera parte de la demostración deducimos que a genera $\mathbb{Z}_{p^{e+1}}^*$. ∎

Del Lema 3.7 se deduce inmediatamente el siguiente:

Teorema 3.8 *Si p es primo impar, entonces $\mathbb{Z}_{p^n}^*$ es cíclico.*

El Teorema anterior falla para $p = 2$ (ver Problema 2).

Demostración del Teorema 3.5. Supongamos que n es un número de Carmichael. Demostramos primero que n es libre de cuadrados. En caso contrario n es múltiplo de p^2, para un primo p. Sea g es un generador de $\mathbb{Z}_{p^2}$. Entonces $o_{p^2}(g) = p(p-1)$, que no es divisor de $n-1$, pues $p \mid n$. Luego $g^{n-1} \not\equiv 1 \mod n$, en contra de que n es un número de Carmichael. Por tanto n es libre de cuadrados. Sea ahora p un divisor primo de n y sea g un generador de $\mathbb{Z}_p^*$. Como n es libre de cuadrados, $\text{mcd}(p, n/p) = 1$. Del Teorema Chino de los Restos

deducimos que existe un entero a tal que $a \equiv g \mod p$ y $a \equiv 1 \mod n/p$. Como $a^{n-1} \equiv 1 \mod n$, se tiene $g^{n-1} \equiv a^{n-1} \equiv 1 \mod p$ y por tanto $p-1 = o_p(g)$ divide a $n-1$.

Recíprocamente, supongamos que n es compuesto y libre de cuadrados y que $p-1$ divide a $n-1$ para todo divisor primo p de n. Entonces $n = p_1 \dots p_k$, con $p_1, \dots, p_n$ primos distintos que cumplen $p_i - 1 \mid n-1$. Si b es una base módulo n, entonces es coprimo con cada p_i y, del Pequeño Teorema de Fermat se tiene que $b^{p_i-1} \equiv 1 \mod p_i$. Como $p_i - 1$ divide a $n-1$ se tiene que $b^{n-1} \equiv 1 \mod p_i$ para todo i. Finalmente deducimos que $b^{n-1} \equiv 1 \mod n$, del Teorema Chino de los Restos. Esto prueba que n es pseudoprimo en todas las bases, es decir, es un número de Carmichael. ∎

Del Teorema 3.5 se deducen algunas restricciones para los números de Carmichael.

Corolario 3.9 *Si n es un número de Carmichael, entonces*

(1) n es impar.

(2) Si p es un divisor primo de n entonces $n \geq p^2$.

(3) El número de divisores primos de n es mayor o igual que 3.

Demostración. Supongamos que n es un número de Carmichael.

(1) Como n es compuesto y libre de cuadrados, es divisible por un primo impar p. Del Teorema 3.5 se deduce que $p-1$ divide a $n-1$. Eso implica que $n-1$ es par, con lo que n es impar.

(2) Si $n < p^2$, entonces $0 < \frac{n}{p} - 1 < p-1$ y, por tanto, $n-1 = \frac{n}{p}(p-1+1) - 1 \equiv \frac{n}{p} - 1 \not\equiv 0 \mod (p-1)$, en contra del Teorema 3.5.

(3) Si n tiene menos de tres divisores primos, entonces $n = pq$ con $p < q$ primos. Entonces $n > p^2$, contradiciendo (2). ∎

3.3 Test de Solovay-Strassen

Definición 3.10 *Sean n y a dos enteros. Decimos que a es un* resto cuadrático módulo n *si existe un entero k tal que $a \equiv k^2 \mod n$.*

Definimos el símbolo de Jacobi $\left(\frac{a}{n}\right)$ *de la siguiente forma:*

- *Si p es primo, entonces*

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{si } p \mid a; \\ 1, & \text{si } p \nmid a \text{ y } a \text{ es resto cuadrático módulo } p; \\ -1, & \text{si } p \nmid a \text{ y } a \text{ no es resto cuadrático módulo } p. \end{cases}$$

  (En este caso $\left(\frac{a}{p}\right)$ también se llama símbolo de Legendre.*)*

- *Si $n = p_1^{e_1} \cdots p_k^{e_k}$, con $p_1, \dots, p_k$ primos entonces*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \cdots \left(\frac{a}{p_k}\right)^{e_k}.$$

Está claro que si $\mathrm{mcd}(a,n)=1$ y a es resto cuadrático módulo n, entonces $\left(\frac{a}{n}\right)=1$. Sin embargo, el recíproco no es cierto. Por ejemplo,

$$\left(\frac{a}{4}\right)=\begin{cases}0, & \text{si } 2\mid a\\ 1, & \text{si } 2\nmid a\end{cases}$$

En particular $\left(\frac{3}{4}\right)=4$, pero 3 no es resto cuadrático módulo 4.

El siguiente lema recoge algunas propiedades elementales del símbolo de Jacobi cuya demostración dejamos al lector como ejercicio.

Lema 3.11 *Sean a,b,n y m números enteros.*

(1) Si $a\equiv b \mod n$, entonces $\left(\frac{a}{n}\right)=\left(\frac{b}{n}\right)$.

(2) $\left(\frac{ab}{nm}\right)=\left(\frac{a}{n}\right)\left(\frac{b}{n}\right)\left(\frac{a}{m}\right)\left(\frac{b}{m}\right)$.

(3) $\left(\frac{ab^2}{n}\right)=\begin{cases}\left(\frac{a}{n}\right), & si\ \gcd(b,n)=1;\\ 0, & si\ \gcd(b,n)\neq 1.\end{cases}$.

Proposición 3.12 *Si p es un primo impar, entonces $\left(\frac{a}{p}\right)\equiv a^{\frac{p-1}{2}} \mod p$.*

Demostración. Si $p\mid a$, entonces $\left(\frac{a}{p}\right)=0\equiv a^{\frac{p-1}{2}} \mod p$. Supongamos pues que p no divide a a. Entonces $a^{p-1}\equiv 1 \mod p$, con lo que utilizando que $\mathbb{Z}_p$ es un cuerpo se deduce que $a^{\frac{p-1}{2}}\equiv \pm 1 \mod p$. Tenemos que demostrar que los cuadrados de $\mathbb{Z}_p^*$ son precisamente las soluciones de $X^{\frac{p-1}{2}}=1$ en $\mathbb{Z}_p^*$.

Los cuadrados no nulos de $\mathbb{Z}_p$ forman la imagen del siguiente homomorfismo

$$\begin{array}{rccc} C:\mathbb{Z}_p^* & \to & \mathbb{Z}_p^* \\ x & \mapsto & x^2 \end{array}$$

cuyo núcleo es el conjunto de las raíces de X^2-1, es decir $\mathrm{Ker}(C)=\{1,-1\}$. Aplicando el Primer Teorema de Isomorfía se tiene $|\mathrm{Im}(C)|=|Z_p^*|/|\mathrm{Ker}(C)|=\frac{p-1}{2}$.

Por otro lado, los elementos que satisfacen $a^{\frac{p-1}{2}}=1$ forman el núcleo del homomorfismo

$$\begin{array}{rccc} f:\mathbb{Z}_p^* & \to & \mathbb{Z}_p^* \\ x & \mapsto & a^{\frac{p-1}{2}} \end{array}$$

Pero los elementos de la imagen de f son raíces de X^2-1, es decir $\mathrm{Im}(f)\subseteq\{1,-1\}$. Además, como $\mathbb{Z}_p^*$ es cíclico de orden $p-1$, los generadores de f no están en el núcleo, con lo que $\mathrm{Im}(f)=\{1,-1\}$. Por tanto $|\mathrm{Ker}(f)|=|\mathbb{Z}_p^*|/|\mathrm{Im}(f)|=\frac{p-1}{2}$. En conclusión, $\mathrm{Im}(C)$ y $\mathrm{Ker}(f)$ son dos subgrupos de $\mathbb{Z}_p^*$ de orden $\frac{p-1}{2}$. Como $\mathbb{Z}_p^*$ es cíclico, solo tiene un subgrupo de orden $\frac{p-1}{2}$, con lo que $\{\text{Cuadrados de } \mathbb{Z}_p^*\}=\mathrm{Im}(C)=\mathrm{Ker}(f)=\{\text{Soluciones de } X^{\frac{p-1}{2}}=1 \text{ en } \mathbb{Z}_p^*\}$. ∎

Definición 3.13 *Se dice que n es* pseudoprimo de Euler *en la base b si $\left(\frac{b}{n}\right) \equiv b^{\frac{n-1}{2}} \mod n$.*

De la Proposición 3.12 se deduce que todo número primo impar es pseudoprimo de Euler en todas las bases. Además el concepto de pseudoprimo de Euler es más fuerte que el de pseudoprimo como muestra la siguiente proposición.

Proposición 3.14 *Si n es pseudoprimo de Euler en la base b entonces también es pseudoprimo en la base b.*

Demostración. Supongamos que n es pseudoprimo de Euler en la base b. Entonces $b^{\frac{n-1}{2}} \equiv \left(\frac{b}{n}\right) = (\pm 1 \mod n)$. Luego $b^{n-1} \equiv (\pm 1)^2 = 1 \mod n$. ∎

La ventaja del concepto de pseudoprimo de Euler es que no existe un concepto similar al de números de Carmichael para pseudoprimos de Euler. Eso es consecuencia del siguiente Teorema.

Teorema 3.15 *Un número impar n es primo si y sólo si es pseudoprimo de Euler en todos las bases módulo n. Además, si n es un número impar compuesto, entonces no es pseudoprimo en al menos la mitad de las bases módulo n.*

Demostración. La condición necesaria es exactamente la Proposición 3.12.

Supongamos que n es un número impar. Utilizando el Lema 3.11 se deduce fácilmente que la aplicación $f : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ dada por

$$f(a) = a^{\frac{n-1}{2}} \left(\frac{a}{n}\right)$$

es un homomorfismo de grupos multiplicativos. El núcleo de esta aplicación es el conjunto E_n de las bases en las que n es pseudoprimo de Euler. Por tanto el cardinal de E_n divide al número de bases módulo n, con lo cual o bien n es pseudoprimo de Euler en todas las bases o bien no es pseudoprimo de Euler en al menos la mitad de las bases.

Supongamos que n es pseudoprimo de Euler en todas las bases y es un número compuesto. Entonces n es un número de Carmichael, por la Proposición 3.14. Sea p un divisor primo de n y sea g entero que genere $\mathbb{Z}_p^*$. Del Teorema 3.5 se deduce que p y n/p son coprimos. Del Teorema Chino de los Restos deducimos que existe un entero a tal que $b \equiv g \mod p$ y $b \equiv 1 \mod \frac{n}{p}$. En particular, b es una base módulo n y $\left(\frac{b}{q}\right) = 1$ para todo divisor primo q de n/p. Como b es un generador módulo p, no es un cuadrado de $\mathbb{Z}_p^*$ y por tanto $\left(\frac{b}{p}\right) = -1$. Luego $\left(\frac{b}{n}\right) = \left(\frac{b}{p}\right) = -1$. Además $b^{\frac{n-1}{2}} \equiv 1 \not\equiv -1 \mod \frac{n}{p}$, pues n es impar. Por tanto $b^{\frac{n-1}{2}} \not\equiv -1 = \left(\frac{b}{n}\right) \mod n$, en contra de que n es pseudoprimo de Euler en todas las bases. ∎

El Teorema 3.15 es la base teórica del Test de Primalidad de Solovay-Strassen.

Algoritmo 8. Test de Primalidad de Solovay-Strassen

ENTRADA: n, un entero positivo impar y k un entero positivo.
Se eligen aleatoriamente $b_1, \ldots, b_k$, bases diferentes módulo n.
Si n no es pseudoprimo de Euler en la base b_i para algún i, entonces
SALIDA: n no es primo.
En caso contrario,
SALIDA: n es primo con una probabilidad $\geq 1 - \frac{1}{2^k}$.

Obsérvese que el Test de Primalidad de Solovay-Strassen es un algoritmo probabilístico de primalidad. De todas formas necesitamos estimar el tiempo de cálculo de este algoritmo. En cada paso del algoritmo hay que decidir si n es pseudoprimo en una base b. Eso implica calcular $(b^{\frac{n-1}{2}} \mod n)$ y $\left(\frac{b}{n}\right)$. Para lo primero ya conocemos un algoritmo de cálculo rápido: la potenciación doblando cuadrados. El resto de la sección lo vamos a dedicar a mostrar un método rápido de cálculo de $\left(\frac{b}{n}\right)$.

Está claro que no podemos utilizar sin más la definición de $\left(\frac{b}{n}\right)$ pues exigiría factorizar n, y si supiéramos factorizarlo este test no tendría sentido.

Empecemos con algunos casos fáciles. Como

$$\left(\frac{b}{n}\right) = 0 \quad \Leftrightarrow \quad \text{mcd}(b, n) \neq 1,$$

y como conocemos un método rápido para calcular el máximo común divisor de b y n, podemos concentrarnos en el caso en que b y n son coprimos, es decir b es una base módulo n. Por otro lado, resulta fácil descomponer b y n como $b = 2^x b_1$ y $n = 2^y n_1$, con b_1 y n_1 impares. Aplicando el Lema 3.11 tenemos

$$\left(\frac{b}{n}\right) = \left(\frac{2}{2}\right)^{xy} \left(\frac{b_1}{2}\right)^{y} \left(\frac{2}{n_1}\right)^{x} \left(\frac{b_1}{n_1}\right) = 0^{xy} \left(\frac{2}{n_1}\right)^{x} \left(\frac{b_1}{n_1}\right)$$

donde, por convenio ponemos $0^0 = 1$. Esto es correcto pues $\left(\frac{b_1}{2}\right) = 1$, y por tanto, si $x = 0$ ó $y = 0$ entonces $\left(\frac{b}{n}\right) = \left(\frac{b_1}{n_1}\right)$. Por tanto, el cálculo de $\left(\frac{b}{n}\right)$ se reduce a dos casos particulares: $\left(\frac{2}{n}\right)$ y $\left(\frac{b}{n}\right)$ con b y n impares y coprimos. La siguiente proposición proporciona un método rápido para calcular $\left(\frac{2}{n}\right)$.

Proposición 3.16 *Si n es impar entonces*

$$\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}} = \begin{cases} 1, & \text{si } n \equiv \pm 1 \mod 8; \\ -1 & \text{si } n \equiv \pm 3 \mod 8. \end{cases}$$

Demostración. En primer lugar, obsérvese que $n-1$ y $n+1$ son pares y uno de los dos es múltiplo de 4. Por tanto $n^2 - 1 = (n-1)(n+1)$ es múltiplo de 8. Luego, si m es otro número impar entonces $n^2m^2 - n^2 - m^2 + 1 = (n^2-1)(m^2-1)$ es múltiplo de 64 lo que implica que

$$\frac{n^2m^2-1}{8} - \frac{n^2-1}{8} - \frac{m^2-1}{8} = \frac{n^2m^2 - n^2 - m^2 + 1}{8} \equiv 0 \mod 2,$$

o lo que es lo mismo

$$\frac{n^2m^2-1}{8} \equiv \frac{n^2-1}{8} + \frac{m^2-1}{8} \mod 2. \tag{3.1}$$

Sea $f : \mathbb{N} \to \{0, 1, -1\}$ la función dada por

$$f(n) = \begin{cases} 0, & \text{si } 2 \mid n \\ (-1)^{\frac{n^2-1}{8}}, & \text{si } 2 \nmid n. \end{cases}$$

De (3.1) se deduce que $f(nm) = f(n)f(m)$ para todo $n, m \in \mathbb{N}$. Como también se verifica $\left(\frac{2}{nm}\right) = \left(\frac{2}{n}\right)\left(\frac{2}{m}\right)$, para demostrar la proposición basta demostrar que se verifica para $n = p$ primo impar.

Supongamos pues que p es un primo impar. Como 8 divide a $p^2 - 1 = |\mathbb{F}_{p^2}^*|$ y $\mathbb{F}_{p^2}^*$ es cíclico, $\mathbb{F}_{p^2}$ tiene un elemento ω de orden 8, o lo que es lo mismo $\omega^4 = -1$. Pongamos

$$G = \sum_{i=0}^{7} f(j)\omega^j = \omega - \omega^3 - \omega^5 + \omega^7 = 2(\omega - \omega^3).$$

Luego

$$G^2 = 4(\omega^2 - 2\omega^4 + \omega^6) = 8.$$

Combinando esto con la Proposición 3.12 deducimos que

$$G^p = G(G^2)^{\frac{p-1}{2}} = 8^{\frac{p-1}{2}} G = \left(\frac{8}{p}\right) G = \left(\frac{2}{p}\right) G. \tag{3.2}$$

Por otro lado, utilizando que $\mathbb{F}_{p^2}$ tiene característica p y, por tanto elevar a p es un automorfismo de $\mathbb{F}_{p^2}$, deducimos que

$$G^p = \left(\sum_{j=0}^{7} f(j)\omega^j\right)^p = \sum_{j=0}^{7} f(j)^p \omega^{jp} = \sum_{j=0}^{7} f(j)\omega^{jp} = \sum_{i=0}^{7} f(ip)\omega^i = f(p)G. \tag{3.3}$$

En la última parte hemos hecho el cambio de variable $i = jp$, y utilizado que $p^2 \equiv 1 \mod 8$. Comparando (3.2) y (3.3) y utilizando que G es invertible en $\mathbb{F}_{p^2}$ deducimos que $f(p) = \left(\frac{2}{p}\right)$, como queríamos. ∎

El método de cálculo de $\left(\frac{n}{m}\right)$ para n y m impares está basado en la Ley de Reciprocidad Cuadrática.

Teorema 3.17 *[Ley de Reciprocidad Cuadrática] Si n y m son dos números naturales impares entonces*

$$\left(\frac{n}{m}\right) = (-1)^{\frac{(n-1)(m-1)}{4}} \left(\frac{m}{n}\right).$$

Demostración. La demostración se divide en dos casos.

Caso I. $n = p$ y $m = q$, primos.

El Teorema es obvio para $p = q$, con lo que suponemos que p y q son dos primos distintos. Si $k = o_q(p)$, entonces el cuerpo $\mathbb{F} = \mathbb{F}_{p^k}$ de orden p^k tiene un elemento ω de orden q.

Ponemos

$$G = \sum_{j=0}^{q-1} \left(\frac{j}{q}\right) \omega^j = \sum_{j=1}^{q-1} \left(\frac{j}{q}\right) \omega^j.$$

Obsérvese que para todo entero i coprimo con q se verifica

$$\sum_{j=0}^{q-1} \omega^{ji} = \sum_{j=0}^{q-1} \omega^j \quad \text{y} \quad G = \sum_{j=0}^{q-1} \left(\frac{ji}{q}\right) \omega^{ji}. \tag{3.4}$$

Esto es debido a que $\{(ji \mod q) : j = 0, 1, \ldots, q-1\} = \{0, 1, \ldots, q-1\}$. Utilizando esto se tiene

$$\sum_{j=0}^{q-1} \omega^{ji} = \begin{cases} q, & \text{si } q \mid i; \\ 0, & \text{en caso contrario.} \end{cases} \tag{3.5}$$

Esto es obvio si $q \mid i$. En caso contrario, $\sum_{j=0}^{q-1} \omega^{ji} = \sum_{j=0}^{q-1} \omega^j =$ Suma de las raíces del polinomio $X^q - 1 = -$ Coeficiente de X^{q-1} en el polinomio $X^q - 1$, que es 0.

Además

$$\sum_{i=0}^{q-1} \left(\frac{i}{q}\right) = 0 \tag{3.6}$$

pues $\mathbb{F}_q^*$ tiene el mismo número de cuadrados que de no cuadrados pues el conjunto de cuadrados de $\mathbb{F}_q^*$ es el único subgrupo de índice 2 de $\mathbb{F}_q^*$.

Ahora utilizando (3.5), (3.6) y las propiedades del símbolo de Jacobi se tiene

$$\begin{aligned}
G^2 &= \left(\sum_{j=1}^{q-1} \left(\tfrac{j}{q}\right) \omega^j\right) \left(\sum_{j=1}^{q-1} \left(\tfrac{-j}{q}\right) \omega^{-j}\right) = \sum_{j=1}^{q-1} \sum_{k=1}^{q-1} \left(\tfrac{-jk}{q}\right) \omega^{j-k} = \left(\tfrac{-1}{q}\right) \sum_{j=1}^{q-1} \sum_{k=1}^{q-1} \left(\tfrac{jk}{q}\right) \omega^{j-k} \\
&= \left(\tfrac{-1}{q}\right) \sum_{j=1}^{q-1} \sum_{i=1}^{q-1} \left(\tfrac{j^2 i}{q}\right) \omega^{j(1-i)} = (-1)^{\frac{q-1}{2}} \sum_{j=1}^{q-1} \sum_{i=1}^{q-1} \left(\tfrac{i}{q}\right) \omega^{j(1-i)} \\
&= (-1)^{\frac{q-1}{2}} \left(\sum_{j=1}^{q-1} \sum_{i=1}^{q-1} \left(\tfrac{i}{q}\right) \omega^{j(1-i)} + \sum_{i=1}^{q-1} \left(\tfrac{i}{q}\right) \omega^{0(1-i)}\right) = (-1)^{\frac{q-1}{2}} \sum_{j=0}^{q-1} \sum_{i=1}^{q-1} \left(\tfrac{i}{q}\right) \omega^{j(1-i)} \\
&= (-1)^{\frac{q-1}{2}} \sum_{i=1}^{q-1} \left(\tfrac{i}{q}\right) \sum_{j=0}^{q-1} \omega^{j(1-i)} = (-1)^{\frac{q-1}{2}} \left(q + \sum_{i=2}^{q-1} \left(\tfrac{i}{q}\right) \left(\sum_{j=0}^{q-1} \omega^{j(1-i)}\right)\right) = (-1)^{\frac{q-1}{2}} q.
\end{aligned}$$

En el salto de la primera a la segunda línea del cálculo anterior, se ha hecho el cambio $k = ji$ (con $j = 1, 2, \ldots, q-1$).

Entonces

$$G^p = (G^2)^{\frac{p-1}{2}} G = (-1)^{\frac{(p-1)(q-1)}{4}} q^{\frac{p-1}{2}} G = (-1)^{\frac{(p-1)(q-1)}{4}} \left(\frac{q}{p}\right) G. \tag{3.7}$$

Por otro lado, utilizando que $\mathbb{F}$ tiene característica p, que $\left(\frac{j}{q}\right) = \left(\frac{jp}{q}\right)\left(\frac{p}{q}\right)$ y (3.4) tenemos

$$G^p = \sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\omega^{jp} = \left(\frac{p}{q}\right)\sum_{j=0}^{q-1}\left(\frac{jp}{q}\right)\omega^{jp} = \left(\frac{p}{q}\right)\sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\omega^{j} = \left(\frac{p}{q}\right)G. \tag{3.8}$$

Como G es invertible en $\mathbb{F}$, comparando (3.7) y (3.8) y teniendo en cuenta que p es impar, obtenemos $\left(\frac{p}{q}\right) = (-1)^{\frac{(p-1)(q-1)}{4}}\left(\frac{q}{p}\right)$, como deseábamos.

Caso II. Caso General.

Pongamos $n = p_1 \dots p_k$ y $m = q_1 \dots q_l$, con cada p_i y cada q_j primos. Reordenando los p_i y q_j si es necesario podemos suponer que $p_i \equiv 1 \mod 4$ si y sólo si $i > k_1$ y $q_j \equiv 1 \mod 4$ si y sólo si $j > l_1$. Eso implica que

$$((p_i-1)(q_j-1) \mod 8) = \begin{cases} 4, & \text{si } i \le k_1 \text{ y } j \le l_1; \\ 0, & \text{en caso contrario.} \end{cases}$$

Por tanto

$$\sum_{i=1}^{k}\sum_{j=1}^{l}\left(\frac{(p_i-1)(q_j-1)}{4} \mod 2\right) = \begin{cases} 0, & \text{si } 2 \mid k_1 l_1; \\ 1, & \text{si } 2 \nmid k_1 l_1. \end{cases}$$

Además, $n \equiv (-1)^{k_1} \mod 4$ y $m \equiv (-1)^{l_1} \mod 4$, con lo que

$$\left(\frac{(n-1)(m-1)}{4} \mod 2\right) = \begin{cases} 0, & \text{si } 2 \mid k_1 l_1; \\ 1, & \text{si } 2 \nmid k_1 l_1 \end{cases}$$

En resumen

$$\frac{(n-1)(m-1)}{4} \equiv \sum_{i=1}^{k}\sum_{j=1}^{l}\frac{(p_i-1)(q_j-1)}{4} \mod 2.$$

y por tanto, aplicando el Caso I, tenemos

$$\begin{aligned} \left(\tfrac{n}{m}\right) &= \prod_{i=1}^{k}\prod_{j=1}^{l}\left(\tfrac{p_i}{q_j}\right) = \prod_{i=1}^{k}\prod_{j=1}^{l}(-1)^{\frac{(p_i-1)(q_j-1)}{4}}\left(\tfrac{q_j}{p_i}\right) = (-1)^{\sum_{i=1}^{k}\sum_{j=1}^{l}\frac{(p_i-1)(q_j-1)}{4}}\prod_{i=1}^{k}\prod_{j=1}^{l}\left(\tfrac{q_j}{p_i}\right) \\ &= (-1)^{\frac{(n-1)(m-1)}{4}}\left(\tfrac{m}{n}\right). \end{aligned}$$

∎

Con ayuda de la Proposición 3.16 y el Teorema 3.17 se puede calcular el símbolo de Jacobi de forma recursiva. Antes de dar el algoritmo veamos un ejemplo ilustrativo en el que se ha utilizado $(2537 \mod 111) = 95$ y $(111 \mod 95) = 16$.

$$\left(\frac{2537}{111}\right) = \left(\frac{95}{111}\right) = (-1)^{\frac{(95-1)(111-1)}{4}}\left(\frac{111}{95}\right) = -\left(\frac{16}{95}\right) = -\left(\frac{1}{95}\right) = -1.$$

Algoritmo 9. Algoritmo para el cálculo del Símbolo de Jacobi

ENTRADA: Una pareja (n, m) de números naturales.
  Si $\text{mcd}(n, m) \neq 1$, entonces
SALIDA: 0.
  $x = 1$.
  Mientras que m es par,
    $m := m/2$.
  Mientras que $m \neq 1$
    $n := (n \mod m)$.
    Mientras que 4 divide a n,
      $n := n/4$.
    Si n es par, entonces $n := n/2$;
      Si además $m \equiv \pm 3 \mod 8$ entonces $x := -x$.
    Si $(n-1)(m-1) \equiv 4 \mod 8$, entonces $x := -x$.
    Intercambiar n y m.
SALIDA: x.

El tiempo de cálculo del algoritmo presentado para las entradas n y m es $O((\log \max(n, m))^3)$ y, por tanto, el tiempo de cálculo del Test de Primalidad de Solovay-Strassen para es $O(k(\log n)^3)$, donde n es el número sobre el que se desea decidir si es primo y k es el número de bases que se prueban.

3.4 Test de Miller-Rabin

Definición 3.18 *Sean n impar y b una base módulo n. Decimos que n es* pseudoprimo fuerte *en la base b si se verifica una de las dos siguientes condiciones con $n - 1 = 2^s t$ y t impar:*

PF1 $b^t \equiv 1 \mod n$.

PF2 Existe $0 \leq r < s$ *tal que* $b^{2^r t} \equiv -1 \mod n$.

Proposición 3.19 *Un número impar primo es pseudoprimo fuerte en todas la bases.*

Demostración. Supongamos que p es primo impar y sea $p - 1 = 2^s t$ con t impar. Como $\mathbb{Z}_p^*$ es cíclico de orden $p - 1$, si b es una base módulo n, entonces $o_p(b) = 2^{s_1} t_1$ con $0 \leq s_1 \leq s$ y $t_1 \mid t$. Entonces se da uno de los siguientes casos:

(1) $s_1 = 0$, en cuyo caso $b^t \equiv 1 \mod p$.

(2) $s_1 \neq 0$ y, por tanto,

$$(b^{2^{s_1-1}t})^2 = b^{2^{s_1}t} \equiv 1 \mod p$$

y

$$b^{2^{s_1-1}t} \not\equiv 1 \mod p,$$

de donde se deduce que

$$b^{2^{s_1-1}t} \equiv -1 \mod p.$$

En ambos casos se verifica que n es pseudoprimo fuerte en la base b. ∎

Proposición 3.20 *Sean n un número impar y b una base de n. Si n es pseudoprimo fuerte en la base b entonces n es pseudoprimo de Euler en la base b.*

Demostración. Pongamos $n-1 = 2^s t$ con t impar y supongamos que n es pseudoprimo fuerte en la base b.

Caso 1. Si $b^t \equiv 1 \mod n$, entonces

$$b^{\frac{n-1}{2}} = b^{2^{s-1}t} \equiv 1 \mod n$$

y como t es impar tenemos

$$\left(\frac{b}{n}\right) = \left(\frac{b}{n}\right)^t = \left(\frac{b^t}{n}\right) = \left(\frac{1}{n}\right) = 1.$$

Caso 2. Supongamos que $b^{2^r t} \equiv -1 \mod n$ con $0 \leq r < s$.

Sea p un divisor primo de n y pongamos $p-1 = 2^{s'}t'$ con t' impar. Entonces $b^{2^r tt'} \equiv -1 \mod n$ y por tanto

$$b^{2^r tt'} \equiv -1 \mod p. \tag{3.9}$$

Por el Teorema de Fermat

$$b^{2^{s'}tt'} \equiv 1 \mod p. \tag{3.10}$$

De (3.9) (3.10) se deduce que $s' > r$. Entonces utilizando la Proposición 3.12 tenemos

$$\left(\frac{b}{p}\right) = \left(\frac{b}{p}\right)^t \equiv (b^{\frac{p-1}{2}})^t = b^{2^{s'-1}tt'} = (b^{2^r tt'})^{2^{s'-(r+1)}} = (-1)^{2^{s'-(r+1)}} \mod p$$

es decir

$$\left(\frac{b}{p}\right) = \begin{cases} 1, & \text{si } s' > r+1; \\ -1, & \text{si } s' = r+1. \end{cases} \tag{3.11}$$

Pongamos $n = p_1 \cdots p_k p_{k+1} \cdots p_l$ con cada p_i primo de forma que $p_i - 1 = 2^{s_i} t_i$ con t_i impar y

$$s_1 = \ldots = s_k = r+1 < s_{k+1}, \ldots, s_l.$$

De (3.11) se deduce que

$$\left(\frac{b}{n}\right) = \prod_{i=1}^{l} \left(\frac{b}{p_i}\right) = (-1)^k.$$

Por otro lado

$$p_i \equiv \begin{cases} 1 \mod 2^{r+2}, & \text{si } i > k; \\ 1 + 2^{r+1} \mod 2^{r+2}, & \text{si } i \leq k. \end{cases}$$

y, por tanto,

$$n \equiv (1+2^{r+1})^k \equiv \begin{cases} 1, & \mod 2^{r+2}, \quad \text{si } k \text{ es par;} \\ 1+2^{r+1} & \mod 2^{r+2}, \quad \text{si } k \text{ es impar.} \end{cases} \tag{3.12}$$

Además, como $2^{s+1} \mid 2^s(t-1) = n-1-2^s$, tenemos que

$$n \equiv 1+2^s \mod 2^{s+1}.$$

Como $r+2 \leq s+1$

$$n \equiv 1+2^s = \begin{cases} 1 & \mod 2^{r+2}, \quad \text{si } r < s-1; \\ 1+2^{r+1} & \mod 2^{r+2}, \quad \text{si } r = s-1. \end{cases} \tag{3.13}$$

De (3.12) y (3.13) se deduce que k es impar precisamente si $r = s-1$. En tal caso

$$\left(\frac{b}{n}\right) = -1 \equiv b^{2^{s-1}t} = b^{\frac{n-1}{2}} \mod n.$$

Y en el caso contrario, es decir, si $r < s-1$ y k es par, entonces

$$\left(\frac{b}{n}\right) = 1 \equiv (-1)^{2^{s-1-r}} = (b^{2^r t})^{2^{s-1-r}} = b^{2^{s-1}t} = b^{\frac{n-1}{2}} \mod n.$$

Por tanto n es pseudoprimo de Euler en la base b. ∎

Combinando el Teorema 3.12 y las Proposiciones 3.19 y 3.20 se obtiene la siguiente consecuencia.

Corolario 3.21 *Un primo impar es primo si y sólo si es pseudoprimo fuerte en todas las bases.*

El Test de Primalidad de Miller-Rabin es, como el de Solovay-Strassen, un test probabilístico de primalidad. En este caso el algoritmo elige un elemento b de $\mathbb{Z}_n^*$ y decide si n es pseudoprimo fuerte en la base b. Si la respuesta es negativa la salida del algoritmo es que n es compuesto. Si n es pseudoprimo fuerte en la base b, se elige otro b al azar y se repite el test hasta que o bien se encuentre una base en la que el número no sea pseudoprimo fuerte o bien se acepte que probablemente el número es primo. Más precisamente el test podría implementarse con el siguiente algoritmo:

Algoritmo 10. Test de Primalidad de Miller-Rabin

ENTRADA: n, un entero positivo impar, y k, un entero positivo.
Se eligen aleatoriamente $b_1, \ldots, b_k$, bases diferentes módulo n.
Si n no es pseudoprimo fuerte en la base b_i para algún i, entonces
SALIDA: n no es primo.
En caso contrario,
SALIDA: n es primo con una probabilidad $\geq 1 - \frac{1}{4^k}$ (salvo que $n = 9$).

De la Proposición 3.20 se deduce que el Test de Primalidad Miller-Rabin es al menos tan efectivo como el de Solovay-Strassen. De hecho, como se sugiere en la última salida del Algoritmo 10, la probabilidad de error en el caso de que la salida sea "primo" es como mucho $1/4^k$, donde k es el número de bases probadas. Esto está justificado por la siguiente proposición.

Teorema 3.22 (Rabin) *Si n es un número compuesto impar diferente de 9 entonces el número de bases en las que n es pseudoprimo fuerte es a lo sumo un cuarto del total de bases módulo n.*

Demostración. Sea

$$M = \{b \in \mathbb{Z}_n^* : n \text{ es pseudoprimo fuerte en la base } b\}$$

La proposición asegura que $|M| \leq \frac{\phi(n)}{4}$.

Consideremos el siguiente conjunto:

$$X = \{h \geq 0 : a^{2^h} \equiv -1 \mod n \text{ para algún entero } a\}$$

Este conjunto no es vacío pues $(-1)^{2^0} \equiv -1 \mod n$. Por otro lado, si $a^{2^h} \equiv -1 \mod n$ entonces a representa un elemento de orden 2^{h+1} en $\mathbb{Z}_n^*$. Luego X está contenido en $\{h : \mathbb{Z}_n^*$ tiene un elemento de orden $2^{h+1}\}$ y, por tanto, X es finito. Fijamos enteros h y a tales que

$$h = \max(X) \quad \text{y} \quad a^{2^h} \equiv -1 \mod n.$$

También ponemos

$$n - 1 = 2^s t \quad \text{y} \quad n = q_1 \dots q_k.$$

con $2 \mid t$ y $q_i = p_i^{e_i}$ y $p_1, \dots, p_k$ primos distintos. Como $a^{2^h} \equiv -1 \mod n$, se tiene que el orden de a en cada $\mathbb{Z}_{p_i}^*$ es 2^{h+1} y por 2^{h+1} divide a $\phi(p_i) = p_i - 1$. Luego $p_i \equiv 1 \mod 2^{h+1}$ para todo y, por tanto $n \equiv 1 \mod 2^{h+1}$, o sea 2^{h+1} divide a $n - 1 = 2^s t$. Luego $h < s$.

Consideremos los siguientes subconjuntos de $\mathbb{Z}_n^*$:

$$\begin{aligned} P &= \{b \in \mathbb{Z}_n^* : n \text{ es pseudoprimo en la base } b\} \\ Q &= \{b \in \mathbb{Z}_n^* : b^{2^h t} \equiv \pm 1 \mod q_i, \text{ para todo } i\} \\ R &= \{b \in \mathbb{Z}_n^* : b^{2^h t} \equiv \pm 1 \mod n\} \end{aligned}$$

Obsérvese que

$$M \subseteq R \subseteq Q \subseteq P \subseteq \mathbb{Z}_n^*.$$

En efecto, las inclusiones $R \subseteq Q$ y $P \subseteq \mathbb{Z}_n^*$ son obvias. Como $h \leq s$, si $b \in Q$ entonces $b^{n-1} = b^{2^s t} \equiv 1 \mod q_i$ para todo i y por tanto $b^{n-1} \equiv 1 \mod n$, es decir $b \in P$. Finalmente, si n es pseudoprimo fuerte en la base b, entonces o bien $b^t \equiv 1 \mod n$, en cuyo caso obviamente $b \in R$, o bien $b^{2^s t} \equiv -1 \mod n$, para algún $0 \leq s < r$. En el segundo caso $s \leq h$ y, por tanto, $b^{2^h t} \equiv \pm 1 \mod n$, es decir $b \in R$.

Además R, Q y P son subgrupos de $\mathbb{Z}_n^*$. Vamos a ver que $[Q : R] \geq 2^{k-1}$. En efecto, por el Teorema Chino de los Restos, para cada subconjunto X de $\mathbb{N}_k = \{1, \dots, k\}$ existe un entero b_X tal que

$$b_X \equiv \begin{cases} 1 \mod q_i, & \text{si } i \in X; \\ a \mod q_i, & \text{si } i \notin X. \end{cases}$$

Entonces $b_X \in Q$. Sean X e Y dos subconjuntos tales que $b_X b_Y^{-1} \in R$. Entonces o bien $b_X^{2^h t} \equiv b_Y^{2^h t} \mod q_i$ para todo i o bien $b_X^{2^h t} \equiv -b_Y^{2^h t}$ para todo i. Eso implica que $X = Y$ ó $\{X, Y\}$ forman una partición de $\mathbb{N}_k$. Luego los 2^{k-1} elementos de la forma b_X con $1 \in X \subseteq \mathbb{N}_k$ están en diferentes clases módulo R y, por tanto, $[Q : R] \geq 2^{k-1}$. En particular $[\mathbb{Z}_n^* : R] \geq 2^{k-1}$ y, por tanto, $|M|/|\mathbb{Z}_n^*| \leq 1/2^{k-1}$, con lo cual la proposición queda demostrada si $k \geq 3$.

Si $k \leq 2$ entonces n no es un número de Carmichael, es decir $P \neq \mathbb{Z}_n^*$ y tenemos $[\mathbb{Z}_n^* : P] \geq 2$, lo que implica que $[\mathbb{Z}_n^* : R] = [\mathbb{Z}_n^* : P][P : R] \geq 2[Q : R] \geq 2^k$ y también queda demostrado el teorema salvo en el caso en que $k = 1$, es decir $n = p^e$, con p primo y $e \geq 2$. En tal caso $\mathbb{Z}_n^*$ es cíclico y P es el subgrupo de $\mathbb{Z}_n^*$ formado por las soluciones de la ecuación $X^{n-1} = 1$. Pero P también está formado por las soluciones de la ecuación $X^d = 1$, donde $d = \text{mcd}(n-1, \phi(n)) = \text{mcd}(p^e - 1, p^{e-1}(p-1)) = p - 1$. Por tanto $|P| = p - 1$, lo que implica que $[\mathbb{Z}_n^* : M] \geq [\mathbb{Z}_n^* : P] = p^{e-1}$. Esto demuestra la proposición, si $e \geq 2$ ó $p \neq 3$. El caso que queda, $e = 2$ y $p = 3$, o equivalentemente $n = 9$, es el excluido. ∎

Obsérvese que el caso excepcional $n = 9$ es de hecho una excepción ya que 9 es pseudoprimo fuerte sólo en las bases 1 y -1, mientras que $\phi(9) = 6$, es decir, 9 es pseudoprimo fuerte en una tercera parte de las bases.

La estimación del Teorema 3.22 es muy modesta. De hecho no es necesario probar con muchas bases en el test de Miller-Rabin para descubrir que un número compuesto lo es. Veamos esto con algunos datos experimentales. Vamos a denotar por ψ_k al primer número compuesto que es pseudoprimo en todas las bases que sean uno de los primeros k primos. La siguiente lista muestra los primeros valores de ψ_k[1]:

$$\begin{aligned} \psi_1 &= 2\,047 \\ \psi_2 &= 1\,373\,653 \\ \psi_3 &= 25\,326\,001 \\ \psi_4 &= 3\,215\,031\,751 \\ \psi_5 &= 2\,152\,302\,898\,747 \\ \psi_6 &= 3\,474\,749\,660\,383 \\ \psi_7 = \psi_8 &= 341\,550\,071\,728\,321 \end{aligned}$$

3.5 Algoritmos deterministas de primalidad con la Hipótesis de Riemann

Del Teorema 3.22 sabemos que si n es pseudoprimo fuerte en más de la cuarta parte de las bases posibles, entonces es primo. Por desgracia esto no proporciona una prueba de primalidad en tiempo polinomial. En esta sección vamos a ver un resultado de Miller que prueba que si fuera cierta la Hipótesis de Riemann Generalizada entonces tanto el Test de Primalidad de Solovay-Strassen como el de Miller-Rabin proporcionaría una algoritmo determinista de primalidad en tiempo polinomial.

[1] G. Jaeschke, On Strong Pseudoprimes to Several Bases. Math. Comput. 61, 915–926, 1993.

Sea $n \in \mathbb{Z}$. Un *carácter* de Dirichlet módulo n es una aplicación $\chi : \mathbb{Z} \to \mathbb{C}$ que es un homomorfismo de los semigrupo multiplicativos, de forma que si $x \equiv y \mod n$, entonces $\chi(x) = \chi(y)$ y $\chi(x) = 0$ precisamente si $\mathrm{mcd}(x, n) \neq 1$. Obsérvese que dar un carácter de Dirichlet módulo n es equivalente a dar un homomorfismo de grupos $\mathbb{Z}_n^* \to \mathbb{C}^*$. Llamaremos carácter de Dirichlet a toda función $\chi : \mathbb{Z} \to \mathbb{C}$ que sea un carácter de Dirichlet módulo n para algún entero n. Se llama *carácter de Dirichlet trivial* al carácter χ_1 dado por $\chi_1(x) = 1$ para todo $x \in \mathbb{Z}$.

Si χ es un carácter de Dirichlet, entonces la *función de Dirichlet* asociada a χ es la función de variable compleja definida por

$$L_\chi(z) = \sum_{n=1}^{\infty} \frac{\chi(n)}{n^z}.$$

Para cada número complejo sea $\mathrm{Re}(z)$ la parte real de z. La función L_χ converge en el semiplano $\mathrm{Re}(z) > 0$. La *Hipótesis de Riemann* para un carácter de Dirichlet χ afirma que si z es un zero de L_χ con $0 < \mathrm{Re}(z) \leq 1$, entonces $\mathrm{Re}(z) = \frac{1}{2}$. La *Hipótesis de Riemann Generalizada* (HRG) asegura que se verifica la Hipótesis de Riemann para todo carácter de Dirichlet. Necesitaremos el siguiente Teorema, cuya demostración se sale de los límites del curso

Teorema 3.23 *(Ankeny-Montgomery) Si se verifica HRG entonces existe un número real positivo C tal que para todo homomorfismo no trivial de grupos abelianos $f : \mathbb{Z}_n^* \to G$ existe un primo $p < C(\log n)^2$ en $\mathbb{Z}_n^*$ tal que $f(p) \neq 1$.*

Teorema 3.24 *Supongamos que se verifica la HRG. Entonces existe un número real positivo C tal que para cada entero impar $n > 1$ se verifica que n es primo precisamente si es pseudoprimo de Euler para toda base $1 \leq b < C(\log n)^2$.*

Demostración. La condición suficiente está clara pues si n es primo entonces es pseudoprimo fuerte en todas las bases (Teorema 3.20). Supongamos que n es compuesto y sea C la constante del Teorema 3.23. Consideramos el homomorfismo $f : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ dado por $f(a) = (a^{\frac{n-1}{2}} \left(\frac{a}{n}\right) \mod n)$. Del Teorema 3.12 se deduce que f no es el homomorfismo trivial y del Teorema 3.23 se deduce que existe un primo $p < C(\log n)^2$ tal que $f(p) \neq 1$, es decir n no es pseudoprimo de Euler en la base p. ∎

Corolario 3.25 *Si se verifica HRG, entonces tanto el Test de Primalidad de Solovay-Strassen como el de Miller-Rabin proporcionan algoritmos deterministas de primalidad en tiempo polinomial.*

Por desgracia para estar seguros de que el test de Miller-Rabin decida de forma determinista la primalidad de un entero impar en tiempo polinomial necesitamos suponer que la HRG se verifica. De hecho sólo necesitamos la HRG para algunos caracteres pero, aún así, sigue siendo un resultado desconocido.

3.6 Los Test de Lucas y de Pratt

Obsérvese que la negación del Problema de Decisión de Primalidad está obviamente en la clase $\mathbb{NP}$, pues un divisor propio de un número proporciona un certificado de que dicho número es compuesto. Sin embargo, no es obvio que el Problema de Decisión de Primalidad esté en la clase $\mathbb{NP}$, ya que no está claro como proporcionar un certificado de que un número es primo. De hecho, salvo que admitamos la Hipótesis de Riemann Generalizada, hasta ahora no hemos visto ningún test ni certificado de primalidad que sea de tiempo polinomial. Hasta el año 1975 no se consiguió demostrar que el Problema de Decisión de Primalidad está efectivamente en la clase $\mathbb{NP}$.[2] En esta sección vamos a ver por un lado un test determinista de primalidad, conocido como el Test de Lucas, que no es de tiempo polinomial y por el otro el Test de Pratt que es un certificado de primalidad de tiempo polinomial. Esto proporcionará una demostración de que el Problema de Decisión de Primalidad está en la clase $\mathbb{NP}$.

Los Test de Lucas y de Pratt parten de una sencilla observación: Un número n es primo si y sólo si $\phi(n) = n-1$ y en tal caso $\mathbb{Z}_n^*$ es cíclico. Por tanto, n es primo si y sólo si $\mathbb{Z}_n^*$ tiene un elemento de orden $n-1$. Obsérvese también que un elemento a de $\mathbb{Z}_n^*$ tiene orden $n-1$ si $a^{n-1} = 1$ y $a^{\frac{n-1}{p}} \neq 1$ para todo divisor primo p de $n-1$. Es decir tenemos la siguiente proposición.

Proposición 3.26 *Sea n un entero positivo impar. Entonces n es primo precisamente si existe $a \in \mathbb{Z}$ tal que*

$$a^{n-1} \equiv 1 \mod n \quad y \quad a^{\frac{n-1}{p}} \not\equiv 1 \mod n \text{ para todo divisor primo } p \text{ de } n-1.$$

En tal caso a es un generador de $\mathbb{Z}_n^*$.

Por tanto, si conocemos cuáles son los primos que dividen a $n-1$, podemos verificar que un entero a satisface las condiciones de la Proposición 3.26. Si encontramos tal entero tendremos un algoritmo efectivo para demostrar que n es primo. Éste es el conocido como el Test de Lucas. Esto no proporciona un algoritmo de tiempo polinomial para resolver el problema de la primalidad, pues requiere encontrar los divisores primos de $n-1$ y descubrir un entero a que satisfaga las condiciones de la Proposición. Sin embargo sí que proporciona un método efectivo para construir primos. La forma de hacer esto es elegir números $p_1, \ldots, p_k$ que sepamos que son primos (tal vez repetidos) y elegir $n = p_1 \ldots p_k + 1$. Entonces el problema que nos encontramos es intentar buscar una base módulo n que satisfaga las condiciones de la Proposición 3.26. Obviamente tenemos que elegir los primos de forma hábil para tener alguna esperanza de que el número pueda ser primo. Por ejemplo, uno de los primos a multiplicar ha de ser 2; si $p_i \neq 3$ para todo i, entonces debemos elegir los p_i de forma que $p_i \equiv -1$ para un número par de valores de i, ya que en caso contrario n sería múltiplo de 3; etc. Una vez que hemos elegido un buen candidato a primo, antes de intentar utilizar el Test de Lucas es conveniente utilizar alguno de los test probabilísticos (Solovay-Strassen ó Miller-Rabin) para evitar perder el tiempo con un número que no es primo. Si un número ha pasado alguno de estos tests tendremos un buen

[2] Vaughan Pratt. Every prime has a succinct certificate. SIAM Journal on Computing, vol.4, pp.214–220. 1975.

candidato a primo. Además tendremos varias bases con respecto a las que n es pseudoprimo, con lo que dichas bases son buenos candidatos a ser generadores de $\mathbb{Z}_n^*$. Claro que si n es primo y b es un generador de $\mathbb{Z}_n^*$, entonces $b^{\frac{n-1}{2}} \equiv -1 \mod n$. Por tanto un buen candidato a generador de $\mathbb{Z}_n^*$ ha de ser un número b tal que $b^{2^{s-1}t} \equiv -1 \mod n$, siendo $n-1 = 2^s t$ con t primo.

¿Cómo podemos estar seguros de que si n es primo encontraremos rápidamente un elemento primitivo módulo n? Como $\mathbb{Z}_n^*$ es cíclico de orden $n-1$, el número de generadores de $\mathbb{Z}_n^*$ es $\phi(n-1)$, por tanto la probabilidad de que un elemento de $\mathbb{Z}_n^*$ elegido al azar sea un generador es $\frac{\phi(n-1)}{n-1} = \prod_{i=1}^{k} \frac{p_i - 1}{p_i}$, si $p_1, \ldots, p_k$ son los primos que dividen a $n-1$. Este cociente puede variar mucho y pues el conjunto $X = \frac{\phi(m)}{m} : m \in \mathbb{N}\}$ es denso en $(0,1)$. Sin embargo, la media de los elementos de X es[3]

$$\frac{3}{\pi^2} \approx 0'3.$$

Por tanto, si efectivamente p es primo, eligiendo al azar elementos de $\mathbb{Z}_n^*$ podremos encontrar rápidamente un elemento de orden $n-1$ de $\mathbb{Z}_n^*$ y de esa forma nos habremos asegurado de que en efecto n es primo.

Vamos a ver ahora como aprovechar la Proposición 3.26 para obtener un certificado de primalidad. Para ello necesitamos introducir algunas definiciones.

Definición 3.27 *Un* par de Fermat *es una pareja de números naturales (a,n) tales que $(a,n) = (1,2)$ ó $2 \leq a < n$ y n es pseudoprimo en la base a. Denotamos por $\mathcal{F}$ el conjunto de los pares de Fermat y definimos la siguiente relación binaria en $\mathcal{F}$*

$$(b,m) \to (a,n) \quad \Leftrightarrow \quad m \mid n-1 \ y \ a^{\frac{n-1}{m}} \not\equiv 1 \mod n.$$

Sea $(a,n) \in \mathcal{F}$. Claramente, el conjunto $\{(b,m) \in \mathcal{F} : (b,m) \to (a,n)\}$ es finito y para toda sucesion de pares de Fermat que cumpla

$$(a_i, n_i) \to (a_{i-1}, n_{i-1}) \to \ldots \to (a_2, n_2) \to (a_1, n_1) \to (a_0, n_0) = (a,n) \tag{3.14}$$

se tiene que $i < n$. Definimos $r(a,n)$ como el mayor i para el que existe una sucesión en $\mathcal{F}$ como en (3.14). Obsérvese que

$$r(a,n) = \begin{cases} 0, & (b,m) \not\to (a,b) \text{ para todo } (b,m) \in \mathcal{F}; \\ \sup\{r(b,m)+1 : (b,m) \in \mathcal{F}, (b,m) \to (a,n)\}, & \text{en caso contrario.} \end{cases}$$

Una *sucesión de Pratt* de un par de Fermat (a,n) es una sucesión de pares de Fermat $(a_1,n_1), \ldots, (a_k,n_k)$ tales que

$$(a_i, n_i) \to (a,n) \text{ para todo } i = 1, \ldots, k$$

y

$$n - 1 = n_1 \cdots n_k.$$

[3] P. Ribenboim, (1996), The New Book of Prime Number Records (3rd ed.), New York: Springer.

Para cada conjunto S de pares de Fermat ponemos

$$\begin{aligned}\Gamma(S) &= \{(1,2)\}\cup \\ &\quad \{(a,n)\in\mathcal{F} : (a,n) \text{ tiene una sucesión de Pratt formada por elementos de } S\}.\end{aligned}$$

Definimos de forma recursiva

$$\Gamma^0 = \{(1,2)\}, \qquad \Gamma^t = \Gamma(\Gamma^0\cup\Gamma^1\cdots\cup\Gamma^{t-1})$$

y ponemos

$$\Gamma^\infty = \bigcup_{t\geq 0}\Gamma^t.$$

Lema 3.28 *Dados conjuntos S y S' de pares de Fermat y $t,t'\geq 0$ se verifica*

(1) Si $S\subseteq S'$, entonces $\Gamma(S)\subseteq\Gamma(S')$.

(2) Si $t\leq t'$, entonces $\Gamma^t\subseteq\Gamma^{t'}$.

(3) $\Gamma^t\subseteq\Gamma(\Gamma^t)$.

(4) $\Gamma(\Gamma^\infty)=\Gamma^\infty$.

Demostración. Ejercicio. ∎

Teorema 3.29 *Para cada par de Fermat (a,n) las siguientes condiciones son equivalentes:*

(1) $(a,n)\in\Gamma^\infty$.

(2) n es primo y a es generador de $\mathbb{Z}_n^*$.

Además, si (a,n) verifica las condiciones anteriores entonces el menor t para el que $(a,n)\in\Gamma^t$ cumple $t<\log_2(n)$.

Demostración. 1 implica 2. Supongamos que $(a,n)\in\Gamma^t$ y razonamos por inducción sobre t. Para $t=0$ es evidente. Supongamos que $t>0$ y la hipótesis de inducción para $r<t$. Podemos suponer que $(a,n)\notin\Gamma^{t-1}$. Entonces existe una sucesión de Pratt de (a,n) formada por elementos $(a_1,n_1),\ldots,(a_k,n_k)$ de Γ^{t-1}. Por hipótesis de inducción cada n_i es primo. Como además $a^{n-1}\equiv 1 \mod n$, $a^{\frac{n-1}{n_i}}\not\equiv 1 \mod n$ y $n-1=n_1\cdots n_k$, por la Proposición 3.26, n es primo y a es un generador de $\mathbb{Z}_n^*$.

2 implica 1. Sea (a,n) un par de Fermat, tal que n es primo y a es un generador de $\mathbb{Z}_n^*$. Razonamos por inducción sobre $r(a,n)$. Supongamos primero que $r(a,n)=0$. Entonces no existe un par de Fermat $(b,m)\to(a,n)$. Vamos a ver que en tal caso $n-1$ es una potencia de 2. En efecto, en caso contrario $n-1$ tiene un divisor primo impar p. Sea b es un generador de $\mathbb{Z}_p$. Entonces (b,p) es un par de Fermat, y como a es generador de $\mathbb{Z}_n^*$, $a^{\frac{n-1}{p}}\not\equiv 1 \mod n$, es decir $(b,p)\to(a,n)$, en contra de que $r(a,n)=0$. Por tanto, $n-1=2^s$. Si $n\neq 2$, entonces

$a^{\frac{n-1}{2}} \not\equiv 1 \mod n$ y por tanto $(1,2) \to (a,n)$, de nuevo una contradicción. Concluimos que $n = 2$ y, por tanto, $a = 1$.

Supongamos ahora que $r(a,n) > 0$ y la hipótesis de inducción. Sea $n - 1 = p_1 \cdots p_k$ con p_i primo, para cada $i = 1, \ldots, k$, y sea a_i un generador de $\mathbb{Z}_{p_i}^*$. Entonces $a^{\frac{n-1}{p_i}} \not\equiv 1 \mod n$, es decir $(a_i, p_i) \to (a,n)$, con lo que $r(a_i,p_i) < r(a,n)$ para todo i. Por hipótesis de inducción $(a_i,p_i) \in \Gamma^\infty$ y como $(a_1,p_1),\ldots,(a_n,p_n)$ es una sucesión de Pratt de (a,n), entonces $(a,n) \in \Gamma(\Gamma^\infty) = \Gamma^\infty$.

Sea $t = \min\{r \geq 0 : (a,n) \in \Gamma^r\}$. Vamos a demostrar que $t < \log_2 n$ por inducción sobre t. Si $t = 0$, entonces $(a,n) = (1,2)$ y el resultado es evidente. Supongamos que $t > 0$ y la hipótesis de inducción. Entonces existe una sucesión de Pratt $(a_1,n_1),\ldots,(a_k,n_k)$ de (a,n) tal que $(a_i,n_i) \in \Gamma^{t-1}$. Además, t es el mínimo para el que $(a,n) \in \Gamma^t$, necesariamente existe un $i_0 \in \{1,\ldots,k\}$ para el que $(a_{i_0},n_{i_0}) \notin \Gamma^r$ para todo $r < t-1$. Reordenando los (a_i,n_i) podemos suponer que dicho $i_0 = 1$. Por la hipótesis de inducción $t - 1 < \log_2(n_1)$. Como n es primo impar y $n = 1 + n_1 \ldots n_k$, o bien $n = 3$, o bien $k \geq 2$. En el primer caso $t = 1 < \log_2(3) = \log_2(n)$ y en el segundo

$$t < \log_2(n_1) + 1 \leq \log_2(n_1) + \log_2(n_2) \leq \log_2(n_1 \cdots n_k) = \log_2(n-1) < \log_2(n).$$

∎

Ejemplos 3.30 Como $\Gamma^0 = \{(1,2)\}$, se tiene que 2 es primo. Las sucesiones de Pratt que se pueden obtener con Γ^0 son repeticiones de $(1,2)$ que serán sucesiones de Pratt para (a,n) si $n = 2^t + 1$ y $a^{\frac{n-1}{2}} \not\equiv 1 \mod n$. Por ejemplo, para $t = 1$, $n = 3$ y, como $2 \not\equiv 1 \mod 3$, entonces $(2,3) \in \Gamma^1$. Para $t = 2$, entonces $n = 5$ y $2^2 = 4 \not\equiv 1 \mod 5$, luego $(2,5) \in \Gamma^1$. Para $t = 3$, tenemos $n = 9$. Como sabemos que 9 no es primo, el Teorema nos asegura que no puede existir a tal que $a^8 \equiv 1 \mod 9$ y $a^4 \equiv 1 \mod 9$. En efecto, como $\varphi(9) = 6$, si $a^8 \equiv 1 \mod 9$, entonces $o_9(a)$ es un divisor de $\mathrm{mcd}(8,6) = 2$, con lo que $a^2 \equiv 1 \mod 9$ y, por tanto $a^4 \equiv 1 \mod 8$.

A partir de $(1,2)$ y $(2,3)$ podemos obtener $2 \cdot 3 + 1 = 7$ y $2 \cdot 9 + 1 = 19$, etc.

Corolario 3.31 *Todo primo p tiene un certificado de primalidad en un tiempo $O(\log(p)^3)$.*

En particular el Problema de Decisión de Primalidad está en la clase $\mathbb{NP}$.

Demostración. Sea p un entero positivo y sea a es un generador de $\mathbb{Z}_p^*$. Por el Teorema 3.29, $(a,p) \in \Gamma^t$ para algún t. Vamos a demostrar por inducción sobre t que hay un algoritmo de tiempo $O((\log^3 p)^3)$ que verifica que $(a,p) \in \Gamma$. Como eso implica que p es primo, a, junto con los datos de entrada del certificado que verifica que (a,p) está en Γ servirán como datos para un certificado que verifique que p es primo.

Si $t = 0$ entonces $(a,n) = (1,2)$ y no hay nada que demostrar. Supongamos que $t > 0$. Entonces existirá una sucesión de Pratt $(a_1,n_1),\ldots,(a_k,n_k) \in \Gamma^{t-1}$ de (a,p) y, por hipótesis de inducción, cada (a,n_i) tiene un certificado de estar en Γ de tiempo $O((\log n_i)^3)$. Es decir existen datos $d_1,\ldots,d_k$ y un algoritmo que verifica que (a_i,n_i) está en Γ en un tiempo $O((\log n_i)^3)$. Ahora el certificado de que (a,p) pertenece a Γ utiliza los datos $d_1,\ldots,d_k$ de la siguiente forma: En primer lugar, para cada i verifica que $(a_i,n_i) \in \Gamma$ con el algoritmo de tiempo

$O((\log n_i)^3)$. Esto tiene un coste en términos de tiempo de $\sum_{i=1}^{k} O(\log n_i)^3 = O(\log(p)^3)$. En segundo lugar verifica que (a,p) es un par de Fermat, es decir que $(a,p) = (1,2)$ ó $2 \leq a < p$ y $a^{p-1} \equiv 1 \mod p$. Finalmente, verifica que $(a_1,n_1),\ldots,(a_k,n_k)$ es una sucesión de Pratt de (a,p). Es decir, comprueba que $p-1 = n_1 \ldots n_k$ y que $a^{\frac{p-1}{n_i}} \not\equiv 1 \mod p$. Cada una de estas comprobaciones tiene un coste de $O(\log(p)^2)$ operaciones y en total hay $O(\log(p))$ comprobaciones que realizar pues como cada $n_i \geq 2$, se tiene que $k \leq \log_2(p)$. De nuevo obtenemos un tiempo de cálculo del orden $O(\log(p)^3)$. ∎

A efectos prácticos el certificado de primalidad de Pratt funcionaría así: La entrada del certificado es una lista $(X_0, X_1, \ldots, X_t)$ donde

$$\begin{aligned}
X_t &= (a,n)\\
X_{t-1} &= ((a_i,n_i))_{i=1,\ldots,k_1}\\
X_{t-2} &= ((a_{i_1,i_2}),n_{i_1,i_2}))_{i_1=1,\ldots,k_1;i_2=1,\ldots,k_{i_1}}\\
X_{t-3} &= ((a_{i_1,i_2,i_3},n_{i_1,i_2,i_3}))_{i_1=1,\ldots,k_1;i_2=1,\ldots,k_{i_1};i_3=1,\ldots,k_{i_1,i_2}}\\
&\ldots\\
X_{t-j} &= ((a_{i_1,\ldots,i_j},n_{i_1,\ldots,i_j}))_{i_1=1,\ldots,k_1;\ldots;i_j=1,\ldots,k_{i_1,\ldots,i_{j-1}}}.
\end{aligned}$$

Donde, si $(a_{i_1,\ldots,i_{j-1}},n_{i_1,\ldots,i_{j-1}}) = (1,2)$ entonces $k_{i_1,\ldots,i_{j-1}} = 0$ y si $t = 0$ entonces $X_0 = (1,2)$.

Con esta entrada el algoritmo primero verifica que todas las parejas $(a_{i_1,\ldots,i_t},n_{i_1,\ldots,i_t})$ que forman X_0 son de la forma $(1,2)$. Si no es así la salida del algoritmo es FALSO. En caso contrario ya sabremos que las parejas de X_1 están en Γ (de hecho en Γ^0). Si $t = 0$, no habría nada más que hacer. Después para la lista $(a,n,n_1,\ldots,n_k)$ y para cada una de las listas de la forma

$$(a_{i_1,\ldots,i_{j-1}},n_{i_1,\ldots,i_{j-1}},n_{i_1,\ldots,i_{j-1},1},\ldots,n_{i_1,\ldots,i_{j-1},k_{i_1,\ldots,i_{j-1}}})$$

se aplica el siguiente algoritmo Si en alguna de ellas la salida es FALSO, entonces la salida de nuestro algoritmo será también FALSO. En caso contrario la salida será VERDADERO.

Algoritmo 11.

ENTRADA: Una lista $((a,p,n_1,\ldots,n_k))$ de enteros positivos.
Si $(a,p) = (1,2)$ ó
$a^{p-1} \equiv 1 \mod p$, $a^{\frac{p-1}{n_1}} \not\equiv 1 \mod p, \ldots, a^{\frac{p-1}{n_k}} \not\equiv 1 \mod p$ y $p-1 = n_1 \cdots n_k$ entonces
  SALIDA: VERDADERO.
En caso contrario
  SALIDA: FALSO.

Obsérvese que si la salida del algoritmo es VERDADERO tendremos que cada pareja de X_i pertenece a Γ^i, pues de forma recursiva hemos visto que tiene una sucesión de Pratt formada por elementos de Γ^{i-1}. Esto certificará que (a,n) está en Γ^t y por tanto a es un elemento de orden $n-1$ de $\mathbb{Z}_n^*$ y en consecuencia que n es primo.

Ejemplo 3.32 Veamos que

$$\begin{aligned}
X_4 &= (5, 833857),\\
X_3 &= ((1,2),(1,2),(1,2),(1,2),(1,2),(1,2),(2,3),(3,43),(2,101)),\\
X_2 &= ((((1,2)),((1,2)),((1,2)),((1,2)),((1,2)),((1,2)),\\
&\quad ((1,2),(2,3),(3,7)),((1,2),(1,2),(2,5),(2,5))),\\
X_1 &= ((((1,2))),(((1,2))),(((1,2))),(((1,2))),(((1,2))),(((1,2))),\\
&\quad (((1,2)),((1,2)),((1,2),(2,3))),(((1,2)),((1,2)),((1,2),(1,2)),((1,2),(1,2)))))\\
X_0 &= (((((1,2)))),((((1,2)))),((((1,2)))),((((1,2)))),((((1,2)))),((((1,2)))),\\
&\quad ((((1,2))),(((1,2))),(((1,2)),((1,2)))),\\
&\quad ((((1,2))),(((1,2))),(((1,2)),((1,2))),(((1,2)),(((1,2))))))
\end{aligned}$$

sirve de certificado de primalidad de 833857. En efecto, todas las parejas de la última lista son iguales a $(1,2)$. Eso verifica el paso 1 del algoritmo. En los siguientes pasos tenemos ejecutar el Algoritmo 11 para las siguientes entradas: $(1,2,2)$, $(2,3,2)$, $(3,7,2,3)$, $(2,5,2,2)$, $(3,43,2,3,7)$, $(2,101,2,2,5,5)$ y $(5,833857,2,2,2,2,2,2,3,43,101)$. La primera pasa el test claramente y todas las demás también pues

$$3-1=2, \quad 2^{3-1}\equiv 1 \mod 3, \quad 2^{\frac{3-1}{2}}\not\equiv 1 \mod 3;$$

$$7-1=2\cdot 3, \quad 3^{7-1}\equiv 1 \mod 7, \quad 3^{\frac{7-1}{2}}\equiv -1\not\equiv 1 \mod 7, \quad 3^{\frac{7-1}{3}}\equiv 2\not\equiv 1 \mod 7$$

$$43-1=2\cdot 3\cdot 7, \quad 3^{43-1}\equiv 1 \mod 43, \quad 3^{\frac{43-1}{2}}\equiv -1 \mod 43$$

$$3^{\frac{43-1}{3}}\equiv 36 \mod 43, \quad 3^{\frac{43-1}{7}}\equiv 41 \mod 43;$$

$$101-1=2^2\cdot 5^2, \quad 2^{101-1}\equiv 1 \mod 101, \quad 2^{\frac{101-1}{2}}\equiv -1 \mod 100, \quad 2^{\frac{101-1}{5}}\equiv 95 \mod 101;$$

$$833857-1=2^6\cdot 3\cdot 43\cdot 101, \quad 5^{833857-1}\equiv 1 \mod 833857,$$

$$5^{\frac{833857-1}{2}}\equiv -1 \mod 833857, \quad 5^{\frac{833857-1}{3}}\equiv 13513 \mod 833857$$

$$5^{\frac{833857-1}{43}}\equiv 694280 \mod 833857, \quad \text{y} \quad 5^{\frac{833857-1}{101}}\equiv 771193 \mod 833857.$$

Esto certifica que 833857 es primo y que 5 genera $\mathbb{Z}^*_{833857}$.

3.7 El Algoritmo AKS

En la Sección 3.6 hemos visto que no se consiguió demostrar que el Problema de Decisión de Primalidad está en la clase $\mathbb{NP}$ hasta el año 1975. De hecho está en la clase $\mathbb{P}$, es decir existe un Test Determinista de Primalidad, pero esto resultó más difícil de demostrar y sólo se consiguió en 2002 por Agrawal, Kayal y Saxena[4]. Este test está basado en el siguiente teorema.

Teorema 3.33 (Agrawal-Kayal-Saxena) *Sean n un entero mayor que 1 y r un número primo. Supongamos que se satisfacen las siguientes condiciones:*

[4]M. Agrawal, N. Kayal, N. Saxena, "PRIMES is in P", Annals of Mathematics 160 (2004), no. 2, 781–793.

(1) n no es divisible por ningún primo menor o igual que r.

(2) $n^i \not\equiv 1 \mod r$ para todo $1 \leq i \leq \log_2(n)^2$.

(3) $(X-j)^n \equiv X^n - j \mod (n, X^r-1)$, para todo $1 \leq j < r-1$.

Entonces n es una potencia de un primo.

Demostración. Supongamos que n cumple las hipótesis 1, 2 y 3. La Hipótesis 2 nos garantiza que existe un divisor primo p de n tal que $p \not\equiv 1 \mod r$. Ponemos $m = \frac{n}{p}$. Vamos a demostrar que n es una potencia de p. Para ello vamos a ver que basta con demostrar la siguiente afirmación:

> **Afirmación**: Existe un grupo cíclico H y parejas de enteros $(i_1, j_1) \neq (i_2, j_2)$ tales que $0 \leq i_1, j_1, i_2, j_2 < \log_n |H|$ y $h^{p^{i_1} m^{j_1}} = h^{p^{i_2} m^{j_2}}$, para todo $h \in H$.

En efecto, si H, (i_1, j_1) y (i_2, j_2) satisfacen las condiciones de la Afirmación y h es un generador de H, entonces

$$|H| = o(h) \mid p^{i_1} m^{j_1} - p^{i_2} m^{j_2}$$

Pero además

$$p^{i_s} m^{j_s} = p^{i_s - j_s} n^{j_s} \leq n^{\max(i_s, j_s)} < n^{\log_n |H|} = |H|, \quad (s = 1, 2)$$

de donde se deduce que

$$p^{i_1} m^{j_1} = p^{i_2} m^{j_2}.$$

Si n no fuera una potencia de p, entonces n tendría un divisor primo $q \neq p$. Analizando la multiplicidad de q en la igualdad anterior tendríamos que $j_1 = j_2$ y por tanto $i_1 = i_2$, una contradicción.

Por tanto, toda la demostración consiste en encontrar un grupo cíclico H y dos parejas de distintas $(i_1, j_1) \neq (i_2, j_2)$ que satisfagan las condiciones de la Afirmación.

Vamos a fijar una raíz r-ésima primitiva de la unidad ζ en una extensión de $\mathbb{Z}_p$ y ponemos $K = \mathbb{Z}_p(\zeta)$. La existencia de ζ está garantizada pues de la Hipótesis 1 se deduce que p no divide a r (ver la Tarea 10). Consideramos el polinomio

$$F = \frac{X^r - 1}{X - 1} = \prod_{i=1}^{r-1} (X - \zeta^i)$$

y el anillo

$$A = \mathbb{Z}[X]/(p, F) = \mathbb{Z}_p[X]/(F).$$

Como $p \not\equiv 1 \mod r$, es decir r no divide a $p-1$, $\mathbb{Z}_p^*$ no tiene elementos de order r, o lo que es lo mismo F no tiene ninguna raíz en $\mathbb{Z}_p$. Recuérdese esto pues será utilizado más adelante. Por otro lado, $F(\zeta) = 0$ y por tanto existe un homomorfismo de anillos dado por

$$\begin{array}{ccc} A & \stackrel{\pi}{\to} & K \\ P(X) + (F) & \mapsto & P(\zeta) \end{array}$$

Sea u un entero coprimo con r y consideremos el homomorfismo de sustitución

$$\begin{array}{ccc} \mathbb{Z}_p[X] & \xrightarrow{S_u} & \mathbb{Z}_p[X] \\ P & \mapsto & P(X^u) \end{array}$$

Para cada $P \in \mathbb{Z}_p[X]$, se tiene que $P \in (F)$ si y sólo si $P(\zeta^i) = 0$ para todo $i = 1, \ldots, r-1$ si y sólo si $P(\zeta^{ui}) = 0$ para todo $i = 1, \ldots, r-1$ si y sólo si $P(X^u) \in (F)$. Esto implica que S_u induce un endomorfismo de anillos

$$\begin{array}{ccc} A & \xrightarrow{\sigma_u} & A \\ P + (F) & \mapsto & P(X^u) + (F) \end{array}$$

Además cada σ_u es invertible. En efecto, si $u \equiv 1 \mod r$ entonces $P(\zeta^{ui}) - P(\zeta^i) = 0$ y por tanto $\sigma_u(P) = P(X^u) + (F) = P + (F)$, es decir $\sigma_u = 1$. Además, $\sigma_u \sigma_v = \sigma_{uv}$. Por tanto, si $uv \equiv 1 \mod r$, entonces $\sigma_u \sigma_v = 1$. Es decir σ_u es un automorfismo de A para todo entero u coprimo con r y la aplicación $u \mapsto \sigma_u$ induce un homomorfismo de grupos

$$\begin{array}{ccc} \mathbb{Z}_r^* & \xrightarrow{\sigma} & \mathrm{Aut}(A) \\ u & \mapsto & \sigma_u \end{array}$$

Ahora es fácil ver que $o(\sigma_u) = o_r(u)$ (indicación: $X^r - 1 \mid X^k - 1 \Leftrightarrow r \mid k$).

Vamos a poner

$$\Delta = \sigma(\mathbb{Z}_r^*) = \text{ Imagen de } \sigma \quad \text{y} \quad \Gamma = \langle \sigma_n, \sigma_p \rangle \leq \mathrm{Aut}(A).$$

Obsérvese que, como A tiene característica p, se verifica

$$\sigma_p(a) = a^p, \quad \text{para todo } a \in A. \tag{3.15}$$

Consideremos ahora los dos siguientes subgrupos de los grupos de unidades de A y K^* respectivamente:

$$G = \{g \in A^* : \sigma_n(g) = g^n\} \leq A^*, \quad H = \pi(G) \leq K^*.$$

Las definiciones de G, Γ y (3.15) implican que para todo $\sigma \in \Gamma$ existe $s \in \mathbb{Z}$ tal que $\sigma(g) = g^s$ para todo $g \in G$. Es evidente que $\sigma_u(G) = G$ para todo $u \in \mathbb{Z}_r^*$ y por tanto σ_u induce un automorfismo de G, para todo $u \in \mathbb{Z}_r^*$. Además, utilizando que σ es un homomorfismo, es fácil demostrar que $\sigma_m(g) = g^m$ para todo $g \in G$ (Ejercicio).

El grupo H es cíclico, por ser un subgrupo del grupo de unidades de un cuerpo finito, y va a ser el grupo cíclico buscado para la Afirmación.

Pongamos $\gamma = |\Gamma|$. Para encontrar la pareja de enteros de la Afirmación usamos que el número de parejas (i, j) tales que $0 \leq i, j \leq \lfloor \sqrt{\gamma} \rfloor$ es $(\lfloor \sqrt{\gamma} \rfloor + 1)^2 > \gamma$. Por el Principio del Palomar[5], existen parejas de enteros $(i_1, j_1) \neq (i_2, j_2)$ tales que

$$0 \leq i_1, i_2, j_1, j_2 \leq \lfloor \sqrt{\gamma} \rfloor \quad \text{y} \quad \sigma_p^{i_1} \sigma_m^{j_1} = \sigma_p^{i_2} \sigma_m^{j_2}.$$

[5]Para que más de γ palomas se ubiquen en un palomar con γ agujeros al menos dos palomas han de compartir un agujero.

Para estos enteros tenemos

$$g^{p^{i_1}m^{j_1}} = \sigma_p^{i_1}\sigma_m^{j_1}(g) = \sigma_p^{i_2}\sigma_m^{j_2}(g) = g^{p^{i_2}m^{j_2}}, \quad \text{para todo } g \in G.$$

Utilizando que π es suprayectivo, tenemos que

$$h^{p^{i_1}m^{j_1}} = h^{p^{i_2}m^{j_2}}$$

para todo elemento $h \in H$.

En resumen, hemos demostrado que existen dos parejas de enteros $(i_1, j_1) \neq (i_2, j_2)$ con $0 \leq i_1, j_1, i_2, j_2 \leq \sqrt{\gamma}$ y $h^{p^{i_1}m^{j_1}} = h^{p^{i_2}m^{j_2}}$ para todo $h \in H$. Para acabar de demostrar la Afirmación, es suficiente que demostremos la siguiente desigualdad:

$$\log_n |H| > \sqrt{\gamma} \tag{3.16}$$

Para demostrar (3.16) vamos a obtener acotaciones por debajo y por arriba del cardinal de G.

Sea $I = \{0, 1, \ldots, r-2\}$. Como $r < p$ podemos identificar I con un subconjunto de $\mathbb{Z}_p$. Por la elección de p se tiene que $r \nmid p-1$ y, por tanto, ningún elemento de I es una raíz r-ésima primitiva de la unidad, o lo que es lo mismo ningún elemento de I es raíz de F. Eso implica que $X - j + (F) \in A^*$ para todo $j \in J$. Como $\sigma_n(X - j + (F)) = X^n - j + (F)$, la Hipótesis 3 significa que $X^n - j + (F) = (X - j + F)^n$ y, por tanto, $X - j + (F) \in G,$ para todo $j \in I$. Para cada subconjunto J de I pongamos

$$g_J = \prod_{j \in J}(X - j) + (F).$$

Utilizando que G es un grupo deducimos que $g_J \in G$ para todo $J \subseteq I$. Además, todos los g_J son distintos. En efecto, si $g_{J_1} = g_{J_2}$ con $J_1, J_2 \subseteq I$ y $j \in J_1 \setminus J_2$, entonces j es raíz de g_{J_1}. Pero la hipótesis por la Hipótesis 1 garantiza que $p > r$ y, por tanto, j no es raíz de g_{J_2}. Luego el polinomio $f = \prod_{j \in J_1}(X - j) - \prod_{j \in J_2}(X - j) \in \mathbb{Z}_p[X]$ es distinto de 0. Sin embargo, su imagen en A es 0, es decir f es un múltiplo no nulo de F. Como F tiene grado $r-1$, el grado de f es al menos $r-1$, y de hecho es exactamente $r-1$. Eso implica que $J_1 = I$ (pues $j \notin J_2$). Como F y f son mónicos del mismo grado y F divide a f, se tiene que $F = f$. Si $J_2 \neq \emptyset$, entonces $F = g_I - g_{J_2}$ tiene una raíz en I, lo que nos lleva a una contradicción (pues F no tiene raíces en $\mathbb{Z}_p$). Por tanto, $J_2 = \emptyset$, con lo que $F = g_I - g_\emptyset = \prod_{j=0}^{r-2}(X - j) - 1$. Comparando los términos independientes deducimos que $1 = -1$, con lo que $p = 2$, en contra de la Hipótesis 1.

En resumen los polinomios $\{g_J : J \subseteq I\}$ proporcionan 2^{r-1} elementos diferentes de G y por tanto

$$|G| \geq 2^{r-1}, \tag{3.17}$$

que es la cota inferior del cardinal de G buscada.

Para obtener la cota superior consideramos un conjunto C de representantes de las clases residuales de Δ módulo Γ y el homomorfismo de grupos

$$\begin{array}{ccc} G & \stackrel{\Pi}{\to} & H^C \\ g & \mapsto & (\pi c(g))_{c \in C} \end{array}$$

Vamos a ver que Π es inyectiva y, por tanto

$$|G| \leq |H|^{[\Delta:\Gamma]}. \tag{3.18}$$

En efecto, sea $g = P(X) + (F) \in \mathrm{Ker}(\Pi)$. Para cada $\sigma \in \Delta$ existen $c \in C$ y $\tau \in \Gamma$ tales que $\sigma = c\tau$. Además existe un entero d tal que $\tau_d(g) = g^d$ para todo $g \in G$. Entonces

$$\pi\sigma(g) = \pi c\tau(g) = \pi c(g^d) = (\pi c(g))^d = 1.$$

Por tanto, para todo $s \in \mathbb{Z}_r^*$ se tiene

$$P(\zeta^s) = \pi(P(X^s) + (F)) = \pi\sigma_s(g) = 1.$$

Es decir, todas las raíces de F son raíces de $P - 1$. Como F no tiene raíces múltiples (¿por qué?), $F \mid P-1$, o sea $g = 1$. Deducimos pues que Π es inyectiva, con lo que queda demostrado (3.18).

Con las acotaciones de $|G|$ obtenidas en (3.17) y (3.18) tenemos

$$2^{r-1} \leq |G| \leq |H|^{[\Delta,\Gamma]} = |H|^{\frac{|\Delta|}{\gamma}} \leq |H|^{\frac{r-1}{\gamma}}$$

y, por tanto

$$2^\gamma \leq |H|.$$

Además, de la Hipótesis 2 del Teorema y de que $\sigma \in \Gamma$ y $|\Gamma| = \gamma$ tenemos

$$\gamma \geq o(\sigma_n) = o_r(n) > (\log_2 n)^2.$$

Luego

$$|H| \geq 2^\gamma = (2^{\sqrt{\gamma}})^{\sqrt{\gamma}} > (2^{\log n})^{\sqrt{\gamma}} = n^{\sqrt{\gamma}},$$

de donde se deduce la desigualdad (3.16) y la demostración está completa. ∎

El Teorema 3.33 proporciona el siguiente algoritmo para decidir si un número es primo.

Algoritmo 12. Test de Primalidad AKS

ENTRADA: Un número positivo impar.

- *Etapa 1*:
  Si n es una potencia de algún número menor que n entonces
  SALIDA: Compuesto.
- *Etapa 2*:
  Calcular el menor número r que no divide a

$$N = 2n(n-1)(n^2-1)\dots(n^{\lfloor(\log_2 n)^2\rfloor}-1).$$

- *Etapa 3*:
  Si n es divisible por algún entero mayor que 1 y menor que r y n entonces
  SALIDA: Compuesto.
- *Etapa 4*:
  Para $b = 1, \dots, r-2$
  Si $(X-b)^n \not\equiv X^n - b \mod (n, X^r-1)$ entonces
  SALIDA: Compuesto.
- *Etapa 5*:
  Si llega a este punto
  SALIDA: Primo.

Vamos a ver que la salida del algoritmo es correcta. Está claro que si sale en las Etapas 1 ó 3, el número es compuesto y la salida es correcta.

Supongamos que la salida se obtiene en la Etapa 4, con lo que el algoritmo nos está diciendo que el número es "compuesto". Esto sucederá porque hay un número b tal que $(X-b)^n$ no es igual a $X^n - b$ en $\mathbb{Z}_n[X]/(X^r-1)$ y, por tanto, dichos polinomios también son diferentes en $\mathbb{Z}_n[X]$. Si n fuera primo, del Pequeño Teorema de Fermat se tendría que $b^n \equiv b \mod n$ y, como $\mathbb{Z}_n[X]$ tiene característica n, también tendríamos $(X-b)^n = X^n - b^n = X^n - b$ en $\mathbb{Z}_n[X]$. Pero $(X-b)^n \neq X^n - b = X^n - b^n$ en $\mathbb{Z}_n[X]$, con lo que tenemos una contradicción. Por tanto, n es efectivamente compuesto.

Finalmente, supongamos que el algoritmo ha llegado a la Etapa 5, y en consecuencia la salida ha sido "primo". Como el algoritmo no ha parado en la etapa 4, se verifica la igualdad $(X-b)^n = X^n - b$ en $\mathbb{Z}_n[X]/(X^r-1)$, para todo $b = 1, \dots, r$. Esto último significa que n satisface la Hipótesis 3 del Teorema 3.33. Como además r ha sido elegido para que no sea divisible por N, entonces n satisface la Hipótesis 2 de dicho Teorema. Además, como el algoritmo no ha parado en la Etapa 3, n no es divisible por ningún número $\leq r$, es decir n satisface la Hipótesis 1 del Teorema. El Teorema implica pues que n es una potencia de un primo, pero como el algoritmo no ha parado en la Etapa 1, sabemos que no es potencia de un número menor y por tanto n es primo. ∎

Para estimar el tiempo de cálculo del Algoritmo AKS se utiliza la siguiente versión débil del Teorema del Número Primo.

Teorema 3.34 *Si n es un número natural, entonces el producto de los primos menores o iguales que $2n^2$ es mayor o igual que 2^n.*

Demostración. Para cada número n, denotamos por $P(n)$ al conjunto de los primos menores o iguales que n. Para cada primo p pongamos $k_p = \sum_{m=1}^{\infty} \left(\left\lfloor \frac{2n}{p^m} \right\rfloor - 2 \left\lfloor \frac{n}{p^m} \right\rfloor \right)$.

Claramente un primo p divide a $n!$ si y sólo si $p \in P(n)$. Además $\{1, 2, \ldots, n\}$ contiene $\left\lfloor \frac{n}{p^m} \right\rfloor$ múltiplos de p^m. Usando eso se deduce que $v_p(n!) = \sum_{m=1}^{\infty} \left\lfloor \frac{n}{p^m} \right\rfloor$ y por tanto

$$v_p\binom{2n}{n} = v_p((2n)!) - 2v_p(n!) = \sum_{m=1}^{\infty} \left\lfloor \frac{2n}{p^m} \right\rfloor - 2 \sum_{m=1}^{\infty} \left\lfloor \frac{n}{p^m} \right\rfloor = k_p. \tag{3.19}$$

Cada $\left\lfloor \frac{2n}{p^m} \right\rfloor - 2 \left\lfloor \frac{n}{p^m} \right\rfloor$ es 0 ó 1 y si es 1 entonces $m \leq \lfloor \log_p(2n) \rfloor$. Por tanto, $k_p \leq \lfloor \log_p(2n) \rfloor = \left\lfloor \frac{\log_2(n)}{\log_2(p)} \right\rfloor$. Combinando esto con (3.19) deducimos que

$$\binom{2n}{n} = \prod_{p \in P(2n)} p^{k_p} \leq \prod_{p \in P(2n)} p^{\lfloor \log_2(2n) \rfloor}.$$

Por otro lado

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} = \prod_{i=1}^{n} \frac{2i(2i-1)}{i^2} = 2^n \prod_{i=1}^{n} \left(2 - \frac{1}{i} \right) \geq 2^n.$$

Combinando las dos últimas desigualdades para n^2, y tomando logaritmos en base 2 obtenemos $n^2 \leq \log_2(2n^2) \log_2 \left(\prod_{p \in P(2n^2)} p \right)$ o lo que es lo mismo

$$n \leq \frac{\log_2(2n^2)}{n} \log_2 \left(\prod_{p \in P(2n^2)} p \right).$$

Consideremos la función $f(x) = \frac{\log_2(2x^2)}{x}$. Es fácil ver que esta función es decreciente para $x > \sqrt{2}$ y $f(4) < 1$. Esto demuestra que para $n \geq 4$ se tiene $n \leq \log_2 \left(\prod_{p \in P(2n^2)} p \right)$, o lo que es lo mismo $\prod_{p \in P(2n^2)} p \geq 2^n$. Esta desigualdad es precisamente el enunciado del teorema y se puede verificar facilmente para $n = 1, 2$ y 3. ∎

Teorema 3.35 *El Algoritmo AKS tiene un tiempo de ejecución $O((\log n)^{22})$ para un número impar n.*

Demostración. Para demostrar el Teorema tenemos que ver que cada etapa tiene un coste en tiempo de ejecución acotado por un polinomio de grado ≤ 22. Por supuesto, no es necesario considerar la Etapa 5.

Etapa 1. En la primera etapa del algoritmo se trata de decidir si $n = a^k$ para algún entero $a < n$. En primer lugar, $a \geq 2$, con lo que si $n = a^k$, entonces $k = \log_a n \leq \log_2 n$. Por tanto un algoritmo muy naïf consiste en recorrer todos los números $2 \leq k \leq \lfloor \log_2 n \rfloor$ y tratar de determinar si $\sqrt[k]{n}$ es un entero o no. Dejamos que el lector muestre cómo se puede hacer esto en tiempo $O((\log n)^3)$. Por tanto, en el peor de los casos esta fase del algoritmo tiene un coste $O((\log n)^4)$.

Etapa 2. Poniendo $m = \lfloor (\log_2 n)^2 \rfloor$ y tomando logaritmos en la igualdad $N = 2n(n-1)(n^2-1)\dots(n^{\lfloor (\log_2 n)^2 \rfloor} - 1)$, obtenemos

$$\log_2 N = 1 + \log_2 n + \sum_{i=1}^{m} \log_2(n^i - 1) \leq 1 + \log_2 n + \sum_{i=1}^{m} i \log_2 n = 1 + \log_2(n) \left(1 + \frac{m(m-1)}{2}\right).$$

Por tanto calcular el valor de N, que es hacer m multiplicaciones de números menores o iguales que N, tiene un coste menor o igual que

$$m(\log_2 N)^2 \leq (\log_2 n)^2 \left(1 + (\log_2 n)\left(1 + \frac{(\log_2 n)^2((\log_2 n)^2 - 1)}{2}\right)\right)^2,$$

que es un polinomio en $\log_2 n$ de grado 12. Además de la primera desigualdad sabemos que existe un entero k tal que

$$\log_2 N < k \leq 2 + \log_2(n) \left(1 + \frac{m(m-1)}{2}\right)$$

En particular $N < 2^k$. Del Teorema 3.34 deducimos que N es menor que el producto de los primos menores o iguales que $2k^2$. Eso implica que el menor primo r que no divide a N es menor o igual que $2k^2 = O((\log n)^{10})$. Por tanto, para encontrar r basta con ir probando los números de longitud menor o igual que el resultado de evaluar $\log n$ en un polinomio de grado 10, hasta encontrar uno que sea primo. Aunque utilicemos un método lento, como la Criba de Eratóstenes, como estos números están acotados por un polinomio de grado 10 en $\log n$, el tiempo de prueba de cada primo es $O((\log n)^2$. En resumen la Etapa 2 tiene un coste en tiempo $O((\log n)^{12})$.

Etapa 3. Tenemos que probar con los números menores o iguales que r y $n-1$ para ver si son divisores de n. Como r es menor o igual que un polinomio de grado 10 en $(\log n)^{10}$, en el peor de los casos el coste de esta etapa es $O((\log n)^{20})$.

Etapa 4. En la última parte del algoritmo tenemos que comparar $(X-b)^n$ y $X^n - b$ en el anillo $A = \mathbb{Z}_n[X]/(X^r - 1)$. En realidad calcular $X^n - b$ en el anillo A, consiste en hacer una división de dos polinomios en $\mathbb{Z}_n[X]$. Dejamos que el lector calcule el coste de este cálculo.

Por otro lado, para calcular $(X-b)^n$ en A, podemos utilizar exponenciación doblando cuadrados. De esta forma, calcular $(X+b)^n$ se reduce a calcular $\log n$ cuadrados en el anillo A. ¿Cuánto cuesta calcular un cuadrado en A? Cada elemento de A está representado por un polinomio $f = f_0 + f_1 X + \dots f_{r-1} X^{r-1}$ de grado menor que r, donde los coeficientes están en $\mathbb{Z}_n$. Por otro lado

$$(f_0 + f_1 X + \dots f_{r-1} X^{r-1})(g_0 + g_1 X + \dots g_{r-1} X^{r-1}) = \sum_{i=0}^{r-1} \left(\sum_{j+k \equiv 1 \mod r} f_j g_k \right) X^i.$$

Por tanto, multiplicar dos de estos polinomios equivale a hacer r operaciones cada una de ellas formada por una suma de r multiplicaciones de elementos de $\mathbb{Z}_n$. Es decir, multiplicar dos elementos de A básicamente consiste en hacer r^2 multiplicaciones de elementos de $\mathbb{Z}_n$, lo que proporciona de nuevo un coste de $O((\log n)^{22})$. Esta es la parte dominante del algoritmo. ∎

En realidad se puede demostrar que el Test de Primalidad AKS tiene un tiempo de ejecución cercano a $O((\log n)^{12})$ pero esto supera los métodos utilizados en este curso. Por desgracia, aunque el Algoritmos AKS sea de tiempo polinomial en términos prácticos no es muy eficiente y de momento para números de 1000 bits es más rápido utilizar el algoritmo que veremos en la siguiente sección. Existe la esperanza de que en los próximos años aparezca una sofisticación del Test de Primalidad AKS que aumente su eficiencia.

3.8 Test de sumas de Gauss

En esta sección vamos a ver el test determinista de primalidad que a efectos prácticos resulta ser el más rápido para los primos de la longitud que a día de hoy es necesario utilizar. En realidad no es un algoritmo de tiempo polinomial sino subexponencial. Más concretamente existe una constante positiva c tal que el tiempo de este algoritmo es $O(k^{c \ln \ln k}) = O(2^{c \ln k \ln \ln k})$ para un número de k-bits. El algoritmo que veremos es una mejora del introducido por Adleman, Rumely y Pomerance[6]. En concreto el algoritmo que veremos fue introducido por Lenstra[7]

Para cada entero positivo n vamos a fijar una raíz compleja n-ésima primitiva de la unidad ζ_n. Por un resultado de Gauss se tiene que el polinomio irreducible de ζ_n sobre $\mathbb{Q}$ es

$$\Phi_n = \prod_{i \in \mathbb{Z}_n^*} (X - \zeta_n^i).$$

En particular, el cuerpo $\mathbb{Q}(\zeta_n)$ es independiente de la raíz n-ésima primitiva de la unidad elegida y es isomorfo a $\mathbb{Q}[X]/(\Phi_n)$ y el anillo $\mathbb{Z}[\zeta_n]$ es isomorfo a $\mathbb{Z}[X]/(\Phi_n)$. Además $[\mathbb{Q}(\zeta_n) : \mathbb{Q}] = \varphi(n)$.

Lema 3.36 *Si $n \nmid m$ y $\zeta_m^j \equiv \zeta_m^k \mod n$ entonces $\zeta_m^j = \zeta_m^k$.*

Demostración. Sin pérdida de generalidad podemos suponer que $k = 0$ y basta demostrar que si $1 \leq j < m$ entonces $\zeta_m^j \not\equiv 1 \mod n$. En efecto, las raíces de $\frac{X^m-1}{X-1}$ son los elementos de la forma ζ_m^j con $j = 1, \ldots, m-1$. Por tanto $\prod_{i=1}^{m-1}(X - \zeta_m^j) = \frac{X^m-1}{X-1} = 1 + X + X^2 + \cdots + X^{m-1}$ y sustituyendo X por 1 tenemos $\prod_{i=1}^{m-1}(1 - \zeta_m^j) = m$. Como $n \nmid m$, deducimos que $\zeta_m^j \not\equiv 1 \mod n$ para todo $j = 1, \ldots, m-1$. ∎

Obsérvese que si n y m son coprimos entonces $\zeta_n\zeta_m$ es una raíz nm-ésima primitiva de la unidad con lo que $\mathbb{Q}(\zeta_n, \zeta_m) = \mathbb{Q}(\zeta_{nm})$. Eso implica que $[\mathbb{Q}(\zeta_n, \zeta_m) : \mathbb{Q}(\zeta_n)] = \frac{[\mathbb{Q}(\zeta_n,\zeta_m):\mathbb{Q}]}{[\mathbb{Q}(\zeta_n):\mathbb{Q}]} =$

[6] L. Adleman, C. Pomerance, R. Rumely, *On distinguishing prime numbers from composite numbers*, Ann. Math. 117 (1983) 173–206.

[7] H. Lenstra Jr., Primality testing (after Adleman, Rumelly and Williams). En Seminar Bourbaki 33 (1980/81), vol. 901 of Lecture Notes in Mathematics, 1981. exp. 576.

$\frac{\varphi(nm)}{\varphi(n)} = \varphi(m)$. De ahí deducimos que el polinomio irreducible de ζ_m sobre $\mathbb{Q}(\zeta_n)$ ha de tener grado $\varphi(m) = [\mathbb{Q}(\zeta_m) : \mathbb{Q}]$ y, por tanto, es igual a Φ_m. Por tanto,

$$\begin{aligned}\mathbb{Q}(\zeta_{nm}) &= \mathbb{Q}(\zeta_n, \zeta_m) \cong \mathbb{Q}(\zeta_n)[Y]/(\Phi_m(Y)) \cong (\mathbb{Q}[X]/(\Phi_n))[Y]/(\Phi_m(Y)) \\ &\cong \mathbb{Q}[X, Y]/(\Phi_n(X), \Phi_m(Y))\end{aligned}$$

y análogamente

$$\mathbb{Z}[\zeta_{nm}] = \mathbb{Z}[\zeta_n, \zeta_m] \cong \mathbb{Z}[X, Y]/(\Phi_n(X), \Phi_m(Y)).$$

Vamos a fijar dos primos distintos p y q. Entonces $\Phi_p(X) = 1 + X + X^2 + \cdots + X^{p-1}$ y $\Phi_q(Y) = 1 + Y + Y^2 + \cdots + Y^{q-1}$ con lo que $\{\zeta_p^i \zeta_q^j : 0 \leq i \leq p-2, 0 \leq j \leq q-2\}$ es una base de $\mathbb{Z}[\zeta_p, \zeta_q]$ sobre $\mathbb{Z}$ y de $\mathbb{Q}[\zeta_n, \zeta_m]$ sobre $\mathbb{Q}$. Además, si n es otro entero entonces $\{\zeta_p^i \zeta_q^j : 0 \leq i \leq p-2, 0 \leq j \leq q-2\}$ también es una base de $\mathbb{Z}[\zeta_p, \zeta_q]/(n)$ sobre $\mathbb{Z}_n$. Además ζ_p y ζ_q están sujetos a las siguientes relaciones:

$$1 + \zeta_p + \zeta_p^2 + \cdots + \zeta_p^{p-1} = 1 + \zeta_q + \zeta_q^2 + \cdots + \zeta_q^{q-1} = 0. \tag{3.20}$$

Usando esto es fácil ver que

$$\sum_{i=0}^{p-1} \zeta_p^{ai} = \begin{cases} p, & \text{si } p \mid a; \\ 0, & p \nmid a. \end{cases} \tag{3.21}$$

Obsérvese que la conjugación compleja deja invariante el anillo $\mathbb{Z}[\zeta_p, \zeta_q]$ y por tanto induce un automorfismo de orden 2 de $A_{p,q}$ dado por $\overline{\zeta_p} = \zeta_p^{-1}$ y $\overline{\zeta_q} = \zeta_q^{-1}$.

Supongamos que además $q \equiv 1 \mod p$ y vamos a denotar por g_q al menor entero positivo que representa un generador de $\mathbb{Z}_q^*$. Consideramos el carácter de Dirichlet módulo q dado por

$$\chi_{p,q}(a) = \begin{cases} 0, & \text{si } q \mid a; \\ \zeta_p^x, & \text{si } a \equiv g^x \mod q. \end{cases}$$

Obsérvese que de (3.21) y $q \equiv 1 \mod p$ se deduce

$$\sum_{a=1}^{q-1} \chi_{p,q}(a) = 0.$$

Definimos la *suma de Gauss* de p y q como el siguiente elemento de $\mathbb{Z}[\zeta_p, \zeta_q]$

$$G(p, q) = \sum_{a=1}^{q-1} \chi_{p,q}(a) \zeta_q^a = \sum_{k=1}^{q-1} \zeta_p^k \zeta_q^{g^k}.$$

Lema 3.37 $G(p, q)\overline{G(p, q)} = q$.

Demostración. Si $a \in \mathbb{Z}_q^*$ vamos a denotar por a^{-1} al inverso de a en $\mathbb{Z}_q^*$. Obsérvese que si $a_1 a_2^{-1} \equiv a \mod q$ entonces $\chi(a_1)\overline{\chi(a_2)} = \chi(a)$ y $a_1 - a_2 \equiv (a-1)a_2 \mod q$. Usando esto, y

haciendo el cambio de variable $a_1 = aa_2$ y $a_2 = m$ en la siguiente cuanta tenemos

$$\begin{aligned} G(p,q)\overline{G(p,q)} &= \sum_{a_1=1}^{q-1}\sum_{a_2=1}^{q-1}\chi(a_1)\overline{\chi(a_2)}\zeta_q^{a_1-a_2} = \sum_{a=1}^{q-1}\chi(a)\sum_{m=1}^{q-1}\zeta_q^{(a-1)m} \\ &= \sum_{a=1}^{q-1}\chi_{p,q}(a)(-1+\sum_{m=0}^{q-1}\zeta_q^{(a-1)m}) = \chi_{p,q}(1)(q-1)+\sum_{a=2}^{q-1}(-1)\chi_{p,q}(a) \\ &= q-\sum_{a=1}^{q-1}\chi_{p,q}(a) = q \end{aligned}$$

∎

Lema 3.38 *Si p, q y n son primos distintos dos a dos con $q \equiv 1 \mod p$ entonces*

$$G(p,q)^{n^{p-1}-1} \equiv \chi_{p,q}(n) \mod n.$$

Demostración. Claramente $\chi_{p,q}(n)\chi_{p,q}(n^{p-1}) = \chi_{p,q}(n)^p = 1$. Como $p \nmid n$, del Pequeño Teorema de Fermat tenemos $n^{p-1} \equiv 1 \mod p$ y, por tanto $\chi_{p,q}(a)^{n^{p-1}} = \chi_{p,q}(a)$ para todo a. Además, como n es primo deducimos que

$$\begin{aligned} G(p,q)^{n^{p-1}} &= \left(\sum_{a=1}^{q-1}\chi_{p,q}(a)\zeta_q^a\right)^{n^{p-1}} \equiv \sum_{a=1}^{q-1}\chi_{p,q}(a)^{n^{p-1}}\zeta_q^{an^{p-1}} = \sum_{a=1}^{q-1}\chi_{p,q}(a)\zeta_q^{an^{p-1}} \\ &= \chi_{p,q}(n)\sum_{a=1}^{q-1}\chi_{p,q}(an^{p-1})\zeta_q^{an^{p-1}} = \chi(p,q)G(p,q). \end{aligned}$$

∎

Teorema 3.39 *Sea P un conjunto finito de números primos y pongamos*

$$\begin{aligned} A &= \prod_{p\in P} p, \\ Q &= \{q \in \mathbb{P} : q-1 \mid A\}, \\ B &= \prod_{q\in Q} q, \\ H &= \{(p,q) \in P\times Q : p \mid q-1\}. \end{aligned}$$

Para cada $p \in P$ y cada $q \in Q$ ponemos

$$\begin{aligned} Q_p &= \{q : (p,q) \in H\}, \\ n^{p-1}-1 &= p^{s_p}u_p, \ \textit{con } p \nmid u_p, \\ P_q &= \{p : (p,q) \in H\}, \\ g_q &= \textit{Menor entero positivo que genera } \mathbb{Z}_q^*. \end{aligned}$$

Sea n un entero positivo menor que B^2. Entonces n es primo si y sólo si $n \in P \cup Q$ o satisface las siguientes condiciones:

(1) $\mathrm{mcd}(AB, n) = 1$,

(2) Para todo $(p, q) \in H$ *existe*

$$m \in \{1, \ldots, s_p\} \text{ con } G(p,q)^{p^m u_p} \equiv \zeta_p^l \mod n \text{ para algún } l \in \{0, 1, \ldots, p-1\}. \quad (3.22)$$

Elegimos $w(p, q) = m$ *mínimo y* $l(p, q) = l$ *satisfaciendo (3.22).*

Para todo $p \in P$ *ponemos*

$$\begin{aligned} w(p) &= \max\{w(p,q) : q \in Q_p\}. \\ q_p &= \min\{q \in Q_p : w(p) = w(p,q)\}, \\ \sum_{i=0}^{p-2}\sum_{j=0}^{q-2} a_{ij}(p)\zeta_p^i \zeta_{q_p}^j &= G(p,q_p)^{p^{w(p)-1}u_p} \mod n. \end{aligned}$$

Para todo $q \in Q$ *definimos* $0 \le l(q) < \prod_{p \in P_q} p$ *de forma que* $l(2) = 0$ *y si* q *es impar entonces*

$$l(q) \equiv l(p,q) \mod p, \text{ para todo } p \in Q_q.$$

Finalmente definimos $0 \le l < B$ *a partir de las congruencias*

$$l \equiv g_q^{l(q)} \mod q$$

para todo $q \in Q$.

(3) para todo $p \in P$ *con* $w(p) \ge 2$ *y tal que* $l(p, q) = 0$ *para todo* $q \in Q_p$*, se verifica que o bien todos los* $a_{ij}(p)$ *son* $0, 1$ *ó* $n-1$ *o existe uno* $a = a_{ij}(p) \notin \{0, 1\}$ *con* $\mathrm{mcd}(a(a-1), n) = 1$.

(4) para todo $q \mid B$ *y todo* $1 \le j < A$ *se tiene que* $\mathrm{mcd}(n, (l^j \mod B))$ *es* 1 *ó* n*;*

Demostración. Está claro que si $n \in P \cup Q$, entonces n es primo por la definición de P y Q. Por tanto, en el resto de la demostración suponemos que $n \notin P \cup Q$.

Primero demostramos que si falla una de las condiciones (1)-(4) entonces es n es compuesto. Esto está claro si falla (1) ó (4). Del Lema 3.38 se tiene que si n es primo entonces s_p satisface la condición (3.22), con lo que si falla (2) entonces n es compuesto. Supongamos que falle (3). Entonces tenemos $p \in P$ con $w(p) \ge 2$ y $l(p, q) = 0$ para todo $q \in Q_p$ de forma que no todos $a_{ij}(p)$ son ni cero ni uno y si cogemos uno $a = a_{ij}(p) \ne 0, 1$ entonces $\mathrm{mcd}(a(a-1), n) > 1$. Eso implica que n es compuesto porque si fuera primo entonces n dividiría a a ó $a-1$ lo cual es imposible pues $1 < a < n$.

Vamos a demostrar el recíproco por reducción al absurdo. Por tanto, suponemos que n es compuesto y satisface las condiciones (1)-(4). Sea r el menor divisor primo de n.

Vamos a demostrar que para todo $p \in P$ tenemos

$$r^{p-1} \equiv 1 \mod p^{w(p)} \quad (3.23)$$

Esto está claro si $w(p) = 1$, por el Teorema de Fermat. Suponemos pues que $w(p) \ge 2$ y distinguimos dos casos. Si $l(p, q) \ne 0$ para algún $q \in Q_p$ entonces, $\zeta_p^{l(p,q)} \ne 1$. Como $r \ne p$,

por (1), del Lema 3.36 se tiene que $\zeta_p^{l(p,q)} \not\equiv 1 \mod r$ y por tanto $G(p,q)^{p^{w(p)}u_p} \equiv \zeta_p^{l(p,q)} \not\equiv 1 \mod r$. Eso implica que el orden multiplicativo $o_r(G(p,q))$ de $G(p,q)$ módulo p es múltiplo de $p^{w(p)+1}$. Además, del Lema 3.38 deducimos que $o_r(G(p,q))$ divide a $p(r^{p-1}-1)$. Por tanto $p^{w(p)}$ divide a $r^{p-1}-1$, que es lo que afirma (3.23). Consideremos ahora el caso en que $l(p,q) = 0$ para todo $q \in Q_p$. Vamos a ver que $G(p,q_0)^{p^{w(p)-1}u_p} \not\equiv 1 \mod r$. En caso contrario $G(p,q_p)^{p^{w(p)-1}u_p} \mod n = 1 + r\sum_{i=0}^{p-2}\sum_{j=0}^{q-2} b_{ij}\zeta_p^i\zeta_{q_p}^j$ y alguna $b_{ij}(p)$ no es cero por la minimalidad de $w(p,q_p) = w(p)$. Eso implica que no todos los coeficientes de $G(p,q_p)^{p^{w(p)-1}u_p}$ mod n son 0 ó 1 y todos son o múltiplos de r o congruentes con 1 módulo r. Esto último implica que $r \leq \text{mcd}(a(a-1),n)$ en contra de la hipótesis (3). Esto demuestra que $p^{w(p)}$ divide a $o_r(G(p,q_p))$. Del Lema 3.38 deducimos que $G(p,q_p)^{r^p-1} \equiv 1 \mod r$, con lo que $o_r(G(p,q_p)) \mid r^p - 1$ y por tanto $p^{w(p)}$ divide a $r^p - 1$, que es lo que queríamos demostrar. Esto acaba la demostración de (3.23).

Sea $p \in P$. Como p es coprimo con u_p, existe un entero c_p tal que $b_p = c_p u_p \equiv 1 \mod p$ y por (3.23) existe un entero a_p tal que $c_p(r^p - 1) = p^{w(p)}a_p$. Luego

$$(r^p - 1)b_p = p^{w(p)}u_p a_p.$$

Fijemos un entero $0 \leq a < A$ tal que

$$a \equiv a_p \mod p$$

para todo $p \in P$.

Entonces

$$G(p,q)^{p^{w(p)}u_p} \equiv \zeta_p^{l(p,q)} = \zeta_p^{l(q)} = \chi_{p,q}(g_q^{l(q)}) = \chi_{p,q}(l) \mod r$$

y, por tanto, usando el Lema 3.38 tenemos

$$\chi_{p,q}(r) = \chi_{p,q}(r)^{b_p} \equiv G(p,q)^{(r^{p-1}-1)b_p} = G(p,q)^{p^{w(p)}u_p a_p} = \chi_{p,q}(l)^a \mod r.$$

Usando el Lema 3.36 deducimos que $\chi_{p,q}(r) = \chi_{p,q}(l^a)$ y por tanto $r \equiv l^a \mod q$ para todo $q \in Q$. Eso implica que $r \equiv l^a \mod B$. Pero como $1 < r \leq \sqrt{n} < B$ deducimos que $1 < r = (l^a \mod B) < n$, en contra de la hipótesis (4). ∎

Dado un entero positivo n vamos a poner

$$\alpha(n) = \prod_{q \in \mathbb{P}, q-1 \mid n} q.$$

El siguiente Teorema es la base de cómo se puede utilizar el Teorema 3.39 para dar un criterio de primalidad para un entero n. Su demostración utiliza herramientas de Teoría Analítica de Números que están fuera del alcance de este libro.

Teorema 3.40 *Existe un número real positivo c tal que para todo número real $x > 16$ el menor entero libre de cuadrados A para el que $\alpha(A) > x$ es menor que* $\ln(x)^{c \ln\ln\ln(x)}$.

La única razón de poner $x > 16$ es para que $\ln\ln\ln(x) > 0$ y no supone ninguna restricción para nuestro objetivo. El Teorema 3.40 nos asegura que la primera parte del siguiente algoritmo se puede completar de forma rápida de forma que conocemos los conjuntos P y Q de forma efectiva y además $A < \ln(x)^{c\ln\ln\ln n}$ para una constante c. Además esto sirve para comprobar que el tiempo de cálculo total del algoritmo es $O(\ln(x)^{c\ln\ln\ln n})$ para una constante c. El Teorema 3.39 asegura que la salida del siguiente algoritmo es correcta.

Algoritmo 13. Test de Primalidad de las Sumas de Gauss

ENTRADA: Un entero positivo impar n.
(1) *A menor número libre de cuadrados tal que $\alpha(A)^2 > n$; el conjunto P de los divisores primos de A, el conjunto Q de divisores primos de $B = \alpha(A)$ y para cada $q \in Q$ un elemento primitivo g_q de $\mathbb{Z}_q$.*
Para ello inicializamos $A = -2$ y realizamos la siguientes cuentas hasta que $B = \alpha(A)^2 > n$.
$A = A + 4$;
Factorizamos A y si A es libre de cuadrados entonces P es el conjunto de los primos que dividen a A, $Q := \{q \in \mathbb{P} : q - 1 \mid A\}$ y $B := \prod_{q \in Q} q$.
• Si en el proceso hemos encontrado que n es uno de los elementos de $P \cup Q$ entonces
SALIDA: n es primo.
• Si $\text{mcd}(AB, n) > 1$ entonces
SALIDA: n es compuesto.
(2) *Comprobación de la hipótesis (2) y cálculo de los parametros*
• $H := \{(p, q) \in P \times Q : p \mid q - 1\}$.
• Para cada $p \in P$ calculamos $Q_p := \{q : (p, q) \in H\}$ y $p^{s_p} u_p := n^{p-1} - 1$, con $p \nmid u_p$
Para cada $q \in Q_p$, si no existe $m \in \{1, \ldots, s_p\}$ que satisface (3.22) entonces
SALIDA: n es compuesto.
$w(p, q) :=$ Menor elemento de $\{1, \ldots, s_p\}$ para el que existe $0 \leq l(p, q) < p$ con

$$G(p, q)^{p^{w(p,q)} u_p} \equiv \zeta_p^{l(p,q)} \mod n.$$

$w(p) := \max\{w(p, q) : q \in Q_p\}$, $g_p := \min\{q \in Q_p : w(p) = w(p, q)\}$.
• Para todo $q \in Q$, $P_q = \{p \in P : (p, q) \in H\}$, $q_q :=$ Menor entero positivo que genera $\mathbb{Z}_q^*$.
(3) *Comprobación de la hipótesis (3)*
Para todo $p \in P$ tal que $w(p) \geq 2$ y $l(p, q) = 0$ para todo $q \in Q_p$ calculamos $\sum_{i=0}^{p-2} \sum_{j=0}^{q-2} a_{ij}(p) \zeta_p^i \zeta_{q_p}^j := G(p, q_p)^{p^{w(p)-1} u_p} \mod n$ y buscamos un $a = a_{ij}(p) > 1$.
Para el primero que encontremos, si $\text{mcd}(a(a-1), n) > 1$ entonces
SALIDA: n es compuesto.
(4) *Comprobación de la hipótesis (4)*
• Ponemos $l(2) = 1$ y para todo $q \in Q$ impar usamos el Teorema Chino de los Restos para encontrar $0 \leq l(q) < \prod_{p \in P_q} p$ con $l(q) \equiv l(p, q) \mod p$ para todo $p \in Q_q$.
• Volvemos a utilizar el Teorema Chino de los Restos para encontrar $0 \leq l < B$ que satisface $l \equiv g_q^{l(q)} \mod q$ para todo $q \in Q$.
• Para todo $j = 1, \ldots, A - 1$. Si $l^j \mod F$ es un divisor no trivial de n entonces
SALIDA: n es compuesto.
Si hemos llegado a este punto:
SALIDA: n es primo.

Para hacer las cuentas del Algoritmo 13 no es necesario trabajar con ζ_p y ζ_q y en consecuencia con aproximaciones de estos números. Alternativamente consideramos ζ_p y ζ_q como símbolos sujetos a las relaciones de (3.20). En otras palabras consideramos $\mathbb{Z}[\zeta_p, \zeta_q]$ como

isomorfo al anillo

$$A_{p,q} = \mathbb{Z}[X,Y]/(1+X+X^2+\cdots+X^{p-1}, 1+Y+Y^2+\cdots+Y^{q-1})$$

a través de las aplicación determinada por $X \mapsto \zeta_p$ e $Y \mapsto \zeta_q$. En consecuencia las cuentas del Algoritmo 13 las realizamos en $A_{p,q}$.

En realidad hacer cuentas en $A_{p,q}$ tiene un coste alto porque estamos trabajando con bases de cardinal $(p-1)(q-1)$ y, por tanto, que estamos realizando operaciones con vectores de esta longitud. Aunque los primos p serán pequeños, los primos q pueden ser bastante grandes. En realidad el algoritmo original de Adleman, Pomerance y Rumely en lugar de trabajar con sumas de Gauss trabaja con sumas de Jacobi que son elementos de $\mathbb{Z}[\zeta_p]$ de la forma

$$J(p,q) = \sum_{m=1}^{q-2} \chi_{p,q}(m^b(m-1)),$$

donde b es el menor entero positivo tal que $(b+1)^2 \not\equiv b^p+1 \mod p$. En la práctica b es bastante pequeño y de hecho se puede demostrar que es menor que $\ln^2(p)$. En cualquier caso la idea principal es la misma que en la del algoritmo visto aunque aparecen algunas complicaciones técnicas que lo hacen algo más difícil de comprender.

3.9 Tareas

(1) Programar la Criba de Eratóstenes para construir los primos menores que un número dado. Transformar el programa para construir otro que decida si un número es primo o no. ¿Hasta qué número funciona esto para decidir sobre la primalidad de un número?

(2) Sea n un entero positivo. Demostrar que $\mathbb{Z}_n^*$ es cíclico si y sólo si $n = 1, 2, 4$ ó p^e ó $2p^e$ con p un primo impar y $e \geq 1$.

(3) Programar el Algoritmo 9 para el cálculo del símbolo de Jacobi.

(4) Programar los Test de Primalidad de Solovay-Strassen y de Miller-Rabin. Comparar experimentalmente el tiempo de ejecución de estos dos test y con el programa de la Tarea 1. Escribir un programa que primero aplique un test probabilístico de primalidad y, después el algoritmo determinista de primalidad de la Tarea 1.

(5) Escribir un programa para estimar el porcentaje de bases en las que un número es pseudoprimo y hacer una gráfica con los resultados. Más concretamente, el programa debe tener como entrada un número n, elegir al azar una cantidad suficiente de bases y comprobar si n es pseudoprimo en dichas bases. La salida del programa es el porcentaje de bases en las que n es pseudoprimo. Analizar los resultados obtenidos estadísticamente probando con una cantidad suficiente de números.

Hacer lo mismo para pseudoprimos de Euler y pseudoprimos fuertes y comparar los resultados.

(6) Sea n un número compuesto impar y p un divisor primo de n. Demostrar

(a) Si n es pseudoprimo en la base b, entonces $b^{\frac{n}{p}-1} \equiv 1 \mod p$.

(b) Si $n = 3p$ con $p > 3$ ó $n = 5p$ con $p > 5$, entonces n no es pseudoprimo en ninguna de las bases 2, 5 y 7.

(7) Encontrar los menores números compuestos que son pseudoprimos en las bases 2, 3 y 5 respectivamente.

(8) Demostrar que si p es primo entonces p^2 es pseudoprimo en la base b si y sólo si $b^{p-1} \equiv 1 \mod p^2$.

(9) Sean p y q dos primos impares distintos y sean $n = pq$ y $d = \text{mcd}(p-1, q-1)$.

(a) Demostrar que $n = pq$ es pseudoprimo en la base b si y sólo si $b^d \equiv 1 \mod n$.

(b) Calcular el número de bases en las que n es pseudoprimo.

(c) Decir cuáles son las bases en las que n es pseudoprimo en el caso en que $q = 2p+1$.

(d) Demostrar que si $q = 2p - 1$ entonces n es pseudoprimo en la mitad de las bases y describir cuáles son estas bases. Demostrar que en tal caso n es pseudoprimo de Euler en la cuarta parte de las bases posibles.

(10) Demostrar que para cualquier número b existen infinitos números compuestos que son pseudoprimos en la base b. (Indicación. Para cada número primo impar p que no divide a $(b-1)b(b+1)$ considerar $n = \frac{b^{2p}-1}{b^2-1}$ y demostrar que $2p$ divide a $n-1$.)

(11) Supongamos que $p_1 = 1 + 6m$, $p_2 = 1 + 12m$ y $p_3 = 1 + 18m$ son números primos. Demostrar que $p_1p_2p_3$ es un número de Carmichael.

(12) Demostrar que 561 es el menor número de Carmichael.

(13) Completar los detalles de la demostración del Teorema 3.33 que se han dejado para el lector.

(14) Demostrar que todo número de Fermat, o sea todo número de la forma $F_n = 2^{2^n} + 1$ es pseudoprimo en la base 2. Utilizar el Test de Miller-Rabin para demostrar que el quinto número de Fermat F_5 es compuesto.

(15) Demostrar que si n es una potencia de un primo impar entonces n es pseudoprimo en una base si y sólo si es pseudoprimo fuerte en dicha base.

(16) Sea $\mathbb{F}$ un cuerpo finito de característica impar. Demostrar que la mitad de los elementos de $\mathbb{F}$ son cuadrados y la otra mitad no lo son. Utilizar esto para obtener un algoritmo de tiempo polinomial probabilístico que proporcione un entero que no sea un resto cuadrático módulo p para un primo impar p dado como entrada. Comparar con la Tarea 1.14.**??**.

(17) Estimar el tiempo de cálculo del algoritmo para comprobar que un número es o no potencia de un número menor utilizando el método explicado en la Etapa 1 de la demostración del Teorema 3.35.

(18) Programar el Test de Primalidad AKS y comparar experimentalmente el tiempo de ejecución con el de cualquiera de los test probabilísticos de primalidad y con el algoritmo de primalidad utilizando la definición.

Capítulo 4

Factorización

Ya sabemos que el problema de romper RSA se reduce en tiempo polinomial al problema de factorizar un número compuesto (que sea producto de dos primos distintos). Es un problema abierto si los dos problemas son equivalentes en tiempo polinomial o en tiempo polinomial probabilístico. Comenzamos este capítulo con el Criptosistema de Rabin para el que sí que se sabe que el problema de romperlo es equivalente en tiempo probabilístico al de la factorización (de ciertos números). En el resto del capítulo vamos a estudiar los principales métodos para factorizar un número compuesto, que está claro que se reduce en tiempo polinomial a la unión de los problemas de decisión de primalidad y de encontrar un factor propio de un número compuesto. Ya hemos tratado el problema de decisión de primalidad en el Capítulo 3, con lo que nos concentraremos en estudiar métodos para encontrar divisores propios de un número compuesto dado.

4.1 El Criptosistema de Rabin

Sea p un número primo y sea c un entero. Nos planteamos el problema de encontrar una raíz cuadrada de c módulo p. Si c es múltiplo de p o $p = 2$ entonces el problema es obvio. Si c no es resto cuadrático módulo p el problema no tendrá solución y podemos descubrir esto calculando $\left(\frac{a}{p}\right)$. Por tanto podemos suponer que p es impar y que c es resto cuadrático módulo p coprimo con p. Utilizando la Tarea 3.16 podemos suponer que disponemos de otro número k que no es resto cuadrático módulo p. Entonces el siguiente algoritmo proporciona una raíz cuadrada de c módulo p, siempre que efectivamente c sea resto cuadrático módulo p y k no lo sea.

Algoritmo 14. Raíces cuadradas módulo un primo

ENTRADA: p, un primo impar, $c, k \in \mathbb{Z}$.
Si $p \mid c$ entonces
  SALIDA: 0.
Si $\left(\frac{c}{p}\right) = -1$ entonces
  SALIDA: c no es resto cuadrático módulo p.
Si $\left(\frac{k}{p}\right) = 1$ entonces
  SALIDA: k es resto cuadrático módulo p.
$p-1 := 2^s t$ con t impar; $b := k^t \mod p$; $r := c^{\frac{t+1}{2}}$, $x_{-1} := \frac{r^2}{c} (= c^t)$.
Para $i = 0, \ldots s-2$:
  $j_i := \begin{cases} 0, & \text{si } x_{i-1}^{2^{s-i-2}} \equiv 1 \mod p; \\ 1, & \text{si } x_{i-1}^{2^{s-i-2}} \equiv -1 \mod p. \end{cases}$
  $x_i := (x_{i-1} b^{2^{i+1} j_i} \mod p)$.
SALIDA: $r b^{\sum_{i=0}^{s-2} 2^i j_i} \mod p$.

Vamos a ver que el algoritmo funciona. Para empezar, como k no es resto cuadrático módulo p, $b = k^t$ tampoco lo será y necesariamente b será un generador del único subgrupo de orden 2^s de $\mathbb{Z}_p^*$. Por tanto, b es una raíz 2^s primitiva de la unidad en $\mathbb{Z}_p^*$, con lo que $b^{2^{s-1}} = -1$. Para que podamos construir j_i es necesario que $x_{i-1}^{2^{s-i-1}} \equiv 1 \mod p$. En efecto, para $i = 0$ tenemos

$$x_{-1}^{2^{s-1}} \equiv c^{2^{s-1} t} = c^{\frac{p-1}{2}} \equiv 1 \mod p,$$

pues c es un resto cuadrático módulo p. Si $1 \leq i \leq s-1$ entonces utilizando la definición de j_{i-1} tenemos

$$x_{i-1}^{2^{s-i-1}} \equiv (x_{i-2} b^{2^i j_i})^{2^{s-i-1}} = x_{i-2}^{2^{s-(i-1)-2}} b^{2^{s-1} j_{i-1}} \equiv x_{i-2}^{2^{s-(i-1)-2}} (-1)^{j_{i-1}} \equiv 1 \mod p.$$

Esto garantiza que el algoritmo se completa y, aplicando la última congruencia para $i = s-1$ deducimos que

$$c \equiv c x_{s-2} \equiv c x_{-1} \prod_{i=0}^{s-2} b^{2^{i+1} j_i} \equiv (r b^{\sum_{i=0}^{s-2} 2^i j_i})^2 \mod p,$$

y, por tanto, la salida $r b^{\sum_{i=0}^{s-2} 2^i j_i}$ del algoritmo es en efecto una raíz cuadrada de c (módulo p).

Obsérvese que podemos simplificar el cálculo de los x_i del algoritmo utilizando la idea de "doblar cuadrados" y que la cuenta final se puede ir realizando durante el algoritmo con la misma idea de doblar cuadrados. En la Tarea 4.14 se pide demostrar que el Algoritmo es de tiempo polinomial. En consecuencia, el cálculo de raíces cuadradas módulo primo es fácil. Sin embargo el cálculo de raíces cuadradas módulo no primos no resulta nada fácil como muestra la siguiente

Proposición 4.1 *Sean p y q dos números primos distintos y $n = pq$. El problema de factorizar n se reduce en tiempo polinomial probabilístico al de calcular raíces cuadradas en $\mathbb{Z}_n$.*

Demostración. Podemos suponer sin pérdida de generalidad que n es impar. Por el Teorema Chino de los Restos tenemos un isomorfismo $f : \mathbb{Z}_p^* \times \mathbb{Z}_q^* \to \mathbb{Z}_n^*$ y Los cuadrados de f serán pues los elementos de la forma $f(a, b)$ con a un cuadrado de $\mathbb{Z}_p^*$ y b un cuadrado de $\mathbb{Z}_b$. Como p y q son impares la mitad de los elementos de $\mathbb{Z}_p^*$ son cuadrados y lo mismo pasa con los elementos de $\mathbb{Z}_q^*$. Por tanto la cuarta parte de los elementos de $\mathbb{Z}_n$ serán cuadrados. Por tanto elegimos un elemento al azar de $\mathbb{Z}_n^*$ y calculamos sus raíces cuadradas, si es que las tiene. Se supone que disponemos de un oráculo que puede calcularlas con lo que también puede decidir si existen y al menos la cuarta parte de las veces obtendremos un cuadrado módulo n. Por tanto, después de un tiempo polinomial probabilístico tendremos un número a coprimo con n y sus raíces cuadradas módulo n. Entonces a es resto cuadrático módulo p y resto cuadrático módulo q. Como p y q son primos impares, a tiene dos raíces cuadradas $\pm b_p$ módulo p y dos raíces cuadradas $\pm b_q$ módulo q. Utilizando de nuevo el isomorfismo f resultará que f tiene cuatro raíces cuadradas módulo n. De hecho, estás cuatro raíces cuadradas x_{00}, x_{01}, x_{10} y x_{11} estarán determinadas por

$$x_{ij} \equiv (-1)^i b_p \mod p \quad \text{y} \quad x_{ij} \equiv (-1)^j b_q \mod q.$$

Eso implica que $\gcd(x_{00} - x_{01}, n) = p$ y $\gcd(x_{10} - x_{11}, n) = q$ con lo que calculando estos máximos comunes divisores habremos encontrado p y q. ∎

Dado un entero positivo n vamos a poner $Q_n = \{a^2 : a \in \mathbb{Z}_n^*\}$. O sea Q_n está formado por los cuadrados de $\mathbb{Z}_n^*$. En el caso en que $p \equiv 3 \mod 4$ existe una forma mucho más rápida de calcular raíces cuadradas módulo p, ya que en tal caso, como $n \equiv a^2 \mod p$ para algún entero p se tiene que

$$(n^{\frac{p+1}{4}})^2 = n^{\frac{p+1}{2}} = nn^{\frac{p-1}{2}} \equiv na^{p-1} \equiv n \mod p.$$

Luego $n^{\frac{p+1}{4}}$ es una raíz cuadrada de n módulo p. Por otro lado todo elemento, si $n^{\frac{p+1}{4}} = a^{\frac{p+1}{2}} = (a^{\frac{p+1}{4}})^2$, lo que implica que $n^{\frac{p+1}{4}}$ es a su vez un resto cuadrático módulo p. Esto demuestra que las aplicaciones siguientes son inversa una de la otra en $Q_p = \{a^2 : a \in \mathbb{Z}_p^*\}$:

$$a \mapsto a^2, \quad a \mapsto a^{\frac{p+1}{4}}.$$

En otras palabras

Lema 4.2 *Si p es un primo con $p \equiv 3 \mod 4$ y a y b son restos cuadráticos módulo p entonces*

$$b \equiv a^2 \mod p \Leftrightarrow a \equiv b^{\frac{p+1}{4}} \mod p.$$

En particular elevar al cuadrado es una biyección en $\mathbb{Z}_p^*$, siempre que $p \equiv 3 \mod 4$. Vamos a ver cómo generalizar esto.

Proposición 4.3 *Sea G un grupo abeliano, sea m un entero positivo y pongamos $G^m = \{g^m : g \in G\}$. Entonces, la restricción de la aplicación $g \mapsto g^m$ a G^m es inyectiva si y solo si G no tiene elementos de orden $p^{v_p(m)+1}$ con p un divisor primo de m.*

Demostración. La aplicación dada, llamémosla f, es un homomorfismo de G^m en si mismo cuyo núcleo está formado por los elementos de la forma g^m tal que $g^{m^2} = 1$. Si g tiene orden $p^{v_p(m)+1}$ con p un divisor primo de m entonces g^m es un elemento no trivial del núcleo de f. Recíprocamente, sea g^m un elemento no trivial del núcleo de f. Entonces el orden n de g divide a m^2 pero no divide a m. Por tanto, n es divisible por un primo p tal que $v = v_p(m) < v_p(n) \leq 2v_p(m)$. Pongamos $n = p^k t$ con $p \nmid t$ y sea $h = g^{p^{k-v-1}t}$. Entonces h tiene orden $p^{v+1} = p^{v_p(m)+1}$ y $h^m \in \ker f$. ∎

Corolario 4.4 *Sea n un entero positivo. Entonces la aplicación $x \mapsto x^2$ define un biyección en $\mathbb{Q}_n$ si y solo si n no es divisible ni por 16 ni por ningún primo p congruente con 1 módulo 4.*

Demostración. Utilizando el Teorema Chino de los Restos tenemos un isomorfismo $\mathbb{Z}_n^* \cong \mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_1^{e_k}}^*$, donde $n = p_1^{e_1} \cdots p_k^{e_k}$. Por la Proposición 4.3, la aplicación no es una biyección si y solo si $\mathbb{Z}_n^*$ tiene un elemento de orden 4 si y solo si algún $\mathbb{Z}_{p_i^{e_i}}^*$ tiene un elemento de orden 4. Si p es impar entonces $\mathbb{Z}_{p_i^{e_i}}$ es cíclico de orden $\varphi(p_i^{e_i}) = p_i^{e_i-1}(p_i - 1)$ y por tanto tiene un elemento de orden 4 si y solo si $p_i \equiv 1 \mod 4$. Como $\mathbb{Z}_8^* \cong \mathbb{Z}_2 \times \mathbb{Z}_2$, $\mathbb{Z}_{16}^* \cong \mathbb{Z}_4 \times \mathbb{Z}_2$ y $\mathbb{Z}_{2^k}^*$ es un cociente de $\mathbb{Z}_{2^{k+1}}^*$, se tiene que $\mathbb{Z}_{2^k}$ tiene un elemento de orden 4 si y solo si $k \geq 4$. Esto acaba la demostración. ∎

Vamos a explicar ahora el *Criptosistema de Clave Pública de Rabin*. La clave privada está formada por una terna $k' = (p, q, a)$ con p y q primos distintos tales que $p \equiv q \equiv 3 \mod 4$ y $a \in \mathbb{Z}_n$ con $n = pq$. La clave pública asociada es $k = (n, a)$. Entonces el conjunto de mensajes en claro para la clave $k = (n, a)$ es

$$M_k = \{m \in \mathbb{Z}_n : 2m + a \in Q_n\}$$

y el conjunto de mensajes descifrables con la clave privada $k' = (p, q, a)$ es

$$C_{k'} = \{c \in \mathbb{Z}_n : a^2 + 4c \in Q_n\}.$$

Se cifra el mensaje m con la clave pública $k = (n, a)$ calculando

$$c_k(m) = m(m + a) \mod n.$$

Para descifrar $c \in C_{k'}$ con la clave privada $k' = (p, q, a)$ calculamos, con la ayuda del Teorema Chino de los Restos, el único elemento de $m_1 \in \mathbb{Z}_n$ que satisface

$$m_1 \equiv \frac{-a + (a^2 + 4c)^{\frac{p+1}{4}}}{2} \mod p \quad \text{y} \quad m_1 \equiv \frac{-a + (a^2 + 4c)^{\frac{q+1}{4}}}{2} \mod q.$$

Vamos a comprobar que si $m \in M_k$ entonces $c = c_k(m) \in C_{k'}$ y $c_{k'}(c) = m$. En efecto, como $c = m(m + a) = m^2 + am$, se tiene que $a^2 + 4c = a^2 + 4m^2 + 4am = (a + 2m)^2 \in Q_n$. Eso implica que $a^2 + 4c$ es cuadrado módulo p y cuadrado módulo q. Además, como p y q son primos distintos y $p \equiv q \equiv 3 \mod 4$, se tiene que n satisface las condiciones del Corolario 4.4.

Por tanto, existe un único $d \in Q_n$ tal que $a^2 + 4c = d^2$. Por hipótesis $a + 2m \in Q_n$ y $a^2 + 4c = a^2 + 4m^2 + 4am = (a + 2m)^2$. Por tanto $d = a + 2m$. Pero esto también implica que $a^2 + 4c \equiv (a + 2m)^2 \mod p$ y $a + 2m \in Q_p$. Por el Lema 4.2, $a + 2m \equiv (a^2 + 4c)^{\frac{p+1}{4}} \mod p$, con lo que $m \equiv \frac{-a+(a^2+4c)^{\frac{p+1}{4}}}{2} \mod p$. Análogamente $m \equiv \frac{-a+(a^2+4c)^{\frac{p+1}{4}}}{2} \mod q$. Por la definición de $m_1 = c_k(c)$ se tiene que $m = m_1$.

Está claro que para romper este criptosistema basta con factorizar $n = pq$. Por otro lado si sabemos romper la clave $k = (n = pq, a)$ entonces dado $c \in c_{k'}$ sabremos encontrar el único $m \in M_k$ tal que $m(m + a) = c$. Por lo visto arriba m y c están ligados por la siguiente regla $a^2 + 4c = (a + 2m)^2$. Es decir sabemos calcular la única raíz cuadrada módulo n de $a^2 + 4c$ que pertenece a Q_n. Tomemos un elemento al azar z en $\mathbb{Z}_n$ y calculemos $c = \frac{z^2-a^2}{4}$. Obsérvese que $a^2 + 4c \in Q_n$, con lo que $c \in C_{k'}$ y por tanto, supuestamente sabemos calcular $m = c_{k'}(c)$ (aunque no conozcamos k'). Eso implica que $(a+2m)^2 = a^2+4c = z^2$. Como cada elemento coprimo con $\mathbb{Z}_n$ tiene cuatro raíces cuadradas módulo n, que son opuestas dos a dos, la probabilidad de que $a + 2m$ y z sean iguales u opuestos módulo n es $\frac{1}{2}$. Por tanto, después de varias pruebas habremos encontrado dos enteros z_1 y z_2 que cumplen $z_1 \not\equiv \pm z_2 \mod n$ pero $z_1^2 \equiv z_2^2 \mod n$. Eso implica que $\gcd(z_1 + z_2, n)$ y $\gcd(z_1 - z_2, n)$ son los dos divisores primos de n. Esto demuestra el siguiente

Teorema 4.5 *Romper el Criptosistema de Rabin para una clave pública (n, a) es equivalente en tiempo polinomial probabilístico al de factorizar n.*

En cierto sentido esto demuestra que el Criptosistema de Rabin puede ser más seguro que RSA pues aunque se sabe que romper RSA se reduce en tiempo polinomial al de factorizar, no se ha demostrado que ambos problemas sean equivalentes en tiempo polinomial probabilístico.

4.2 Divisores pequeños

Por supuesto que lo primero que hay que hacer para encontrar un divisor propio de un número compuesto es buscar divisores entre los números primos pequeños. Para eso es conveniente disponer de una lista de los primeros números primos. Esta lista se puede construir utilizando la Criba de Eratostenes como explicamos en la Sección 3.1. En esa sección construimos la lista de los primos menores que 10^7 y ahora podemos utilizarla para buscar divisores del siguiente número:

$$n = 48798798798798798798287946898720987987987909809809 8.$$

Para ello utilizamos el siguiente programa que intenta factorizar el número dado como primer argumento con la lista de primos dada como segundo argumento:

```
DivPequeno := function(n,primos)
local m, factors, l, i, p ;
m:=n; factors := []; l:=664580; i:=1; p:=2;
while p^2<=m and i<=l do
  p:=primos[i];
  if RemInt(m,p)=0 then
```

```
    Add(factors,p);
    m := m/p;
  else
    i:=i+1;
  fi;
od;
Add(factors,m);
return factors;
end;
```

Aplicando `DivPequeno` a n con la lista de primos `ListaPrimos` obtenemos el siguiente resultado.

```
gap> DivPequeno(n,ListaPrimos);
[ 2, 23, 2130173, 49800812053659490279701457702303570799983 31 ]
```

Es decir

$$n = 2 \cdot 23 \cdot 2130173 \cdot 4980081205365949027970145770230357079998331$$

donde los tres primeros divisores son primos y lo único que sabemos del último

$$m = 4980081205365949027970145770230357079998331$$

es que no es divisible por ningún primo menor o igual que 10^7. Antes de intentar factorizar un número es conveniente asegurarse de que no es primo. Para eso es conveniente utilizar alguno de los test probabilísticos vistos en el Capítulo 3. Por ejemplo,

$$2^{m-1} \equiv 24609014816871853214589163387146007204013 65 \not\equiv 1 \mod m,$$

es decir, m no es pseudoprimo en la base 2 y por tanto m no es primo. Sin embargo esto no proporciona ningún divisor propio de m.

Por supuesto que uno de los test probabilísticos rápidos podría no resolver de forma determinística si el número es primo. En tal caso podemos utilizar el test AKS para decidir determinísticamente si el número en cuestión es primo. Por desgracia, hoy por hoy el test AKS es demasiado lento y resulta más eficiente utilizar el Algoritmo de Primalidad de Adleman-Pomerance-Rumely y Cohen-Lenstra que mencionamos en la Sección 3.5.

4.3 Factorización de Fermat

Obsérvese que si d es un divisor propio de n, entonces $\min\{d, n/d\}$ es menor o igual que $\sqrt{n}$. Por tanto, a la hora de buscar divisores propios de n nos podemos restringir a buscar en el intervalo $[1, \sqrt{n}]$. El método de factorización de la sección anterior encontrará divisores pequeños. El método que vamos a ver en esta sección será útil para buscar divisores cercanos a $\sqrt{n}$.

Si x e y son dos números naturales con $x > y+1$ y $n = x^2 - y^2$, entonces $n = (x-y)(x+y)$ es una factorización propia de n. Por otro lado, si n es impar todas las factorizaciones de n se obtienen de esta forma. En efecto, si $n = ab$, entonces $a \equiv b \equiv 1 \mod 2$ y

$$\left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2 = ab = n.$$

Es decir, el problema de encontrar un divisor propio de un número impar es equivalente al de escribirlo como diferencia de dos cuadrados. El siguiente algoritmo proporcionará el mayor divisor propio de n en el intervalo $[1, \sqrt{a}]$.

Algoritmo 15. Algoritmo para calcular el mayor divisor propio

ENTRADA: Un número natural impar n.
$a = \lceil \sqrt{n} \rceil$
Mientras que $b = \sqrt{a^2 - n} \notin \mathbb{N}$,
$a := a + 1$.
SALIDA: $a - b$, el mayor divisor propio de n en el intervalo $[1, \sqrt{a}]$.

Ejemplo 4.6 Sea $n = 3953$, entonces $\lceil \sqrt{3953} \rceil = 63$ y $63^2 - 3953 = 16 = 4^2$, con lo que $3953 = 63^2 - 4^2 = 59 \cdot 67$.

Pongamos $n = 33319$. Entonces $a_0 = \lceil \sqrt{n} \rceil = 183$ y tenemos $183^2 - 33319 = 170$, $184^2 - 33319 = 537$, $185^2 - 33319 = 906$, $186^2 - 33319 = 1277$, $187^2 - 33319 = 1650$, $188^2 - 33319 = 2025 = 45^2$. Por tanto $33319 = (188 - 45)(188 + 45) = 143 \cdot 233$.

Una versión alternativa de este método consiste aplicárselo a kn para algún k pequeño. Supongamos que hemos obtenido la siguiente factorización propia de kn: $kn = a^2 - b^2 = (a - b)(a + b)$. Como k es pequeño y $a + b$ es grande, $a + b$ no divide a k y por tanto $\mathrm{mcd}(n, a + b) \neq 1$.

Algoritmo 16. Factorización de Fermat

ENTRADA: Un número natural impar n y un entero $k \ll \sqrt{n}$.
$a = \left\lceil \sqrt{kn} \right\rceil$
Mientras que $b = \sqrt{a^2 - kn} \notin \mathbb{N}$,
$a = a + 1$.
SALIDA: $\mathrm{mcd}(n, a + b)$.

Ejemplo 4.7 Sea $n = 33491$. Después de cansarnos de aplicar el primer método para n, probamos con $m = 3n = 100473$, tenemos $\lceil \sqrt{m} \rceil = 317$ y $317^2 - m = 16 = 4^2$. Por tanto $3n = 317^2 - 4^2 = 313 \cdot 321$, con lo que $n = 313 \cdot 107$.

El método de factorización de Fermat encontrará rápidamente factores de n cercanos a $\sqrt{n}$ y la segunda versión encontrará factores cercanos a $\sqrt{kn}$, para algún k. La moraleja es que no podemos elegir dos primos demasiado próximos como clave privada en RSA ni primos que sean uno cercano a un múltiplo pequeño del otro.

Sin embargo si un número n no tiene divisores pequeños ni divisores cercanos a $\sqrt{n}$ el método de Fermat puede llevar demasiado tiempo. El siguiente programa es una implementación del Método de Factorización de Fermat en el que el segundo argumento es un limitador del número de intentos.

```
FactFermatMax := function(n,max)
local a,b,i;
a := RootInt(n,2); if a^2<>n then
  a:=a+1;
fi; for i in [1..max] do
  b:=RootInt(a^2-n);
  if n=a^2-b^2 then
    return a-b;
  else
    a:=a+1;
  fi;
od;
return fail;
end;
```

Al aplicar este programa con $m = 4980081205365949027970145770230357079998331$, que ya sabemos que es compuesto, y diez millones de intentos obtenemos una "derrota" después de casi minuto y medio de cálculos.

```
gap> FactFermatMax(m,10^7);
fail
gap> time;
86609
```

4.4 Los métodos de Pollard

Factorización $(p-1)$ de Pollard

Sea n un número compuesto y sea p un factor primo de n que queremos descubrir. Si m es un múltiplo de $p-1$ y a es coprimo con p entonces, del Pequeño Teorema de Fermat se deduce que p divide a $\mathrm{mcd}(a^m-1, n)$. Por tanto, si elegimos m de forma que podamos garantizar que hay una probabilidad alta de que m sea múltiplo de $p-1$, entonces descubriremos un divisor de n, múltiplo de p y, con un poco de suerte, ese divisor será menor que n. Por ejemplo, si los divisores primos de $p-1$ son pequeños, será bastante probable que $p-1$ divida a $k!$, sin que k sea excesivamente grande. El Método $p-1$ de Pollard consiste en ir calculando $\mathrm{mcd}(a^{k!}-1, n)$

para valores cada vez más grandes de k, con la esperanza de encontrar de esta forma un divisor propio de n. Como el primo p es desconocido, no podemos verificar que a sea coprimo con p. Lo que se hace en tal caso es elegir a, coprimo con n, es decir a es una base módulo n.

Algoritmo 17. Factorización $p-1$ de Pollard

ENTRADA: n, a.
$d := \text{mcd}(n, a)$.
• Si $1 < d < n$ entonces,
SALIDA: d.
• Si $d = n$ entonces
SALIDA: Los dos datos de entrada han de ser coprimos.
• $m := a; d := 1; i := 1;$
Mientras que $d = 1$
$m := (m^i \mod n)$; $d := \text{mcd}(m-1, n)$; $i := i+1$;
Si d=n entonces
SALIDA: No se ha encontrado ningún divisor propio.
Si $d \neq 1$ entonces
SALIDA: d.

Por ejemplo, volvamos a la factorización

$$n = 48798798798798798798287946898720987987987909809809 8 = 2 \cdot 23 \cdot 2130173 \cdot m$$

con $m = 4980081205365949027970145770230357079998331$, con la que habíamos comenzado el capítulo. Recordemos que sabíamos que m es un número compuesto que no es divisible por ningún primo menor que 10^7. Sin embargo, hemos aplicado el Algoritmo 15 y, nos hemos cansado de esperar sin que el programa proporcionara ningún divisor propio. Entonces aplicamos el Algoritmo 17 a m con la semilla $a = 2$ y descubrimos que $d = 10534517317$ es un divisor propio de m. De hecho, como n no es múltiplo de ningún primo menor o igual que 10^7 tampoco lo es d. Eso implica que d es primo pues $\sqrt{d} < 10^7$. Por otro lado aplicando el Test de Miller-Rabin obtenemos que m/d es probablemente primo. Por tanto, la siguiente será una factorización de n, probablemente en producto de factores primos, en la que el único factor sobre el que no tenemos certeza de que sea primo es el último:

$$\begin{aligned} n &= 48798798798798798798287946898720987987987909809809 8 \\ &= 2 \cdot 23 \cdot 2130173 \cdot 10534517317 \cdot 4727393819296665386999568183661 43. \end{aligned}$$

Por supuesto que el método podría no haber descubierto ningún divisor propio. En tal caso es una buena idea probar con otra semilla.

Factorización ρ de Pollard

Como hemos visto al principio del capítulo, el primer método elemental de factorización de un número entero compuesto n consiste en buscar divisores entre los primos menores que él, es

decir, se intenta dividir n entre $2, 3, 5, \ldots$. Si no queremos depender de un fichero de primos simplemente intentaremos dividir n por 2 y después intentamos dividir por los números impares empezando por 3 y aumentando el divisor en dos unidades cada vez. El menor factor de n ha de ser menor o igual que $\sqrt{n}$. Si este primer factor es muy grande este método puede llevar mucho tiempo. ¿Y si en lugar de buscar divisores por orden los buscamos al azar? O mejor: ¿Porqué no elegimos números m al azar y calculamos $\mathrm{mcd}(n, m)$ con la esperanza de que el resultado sea un divisor propio de n? Una variación de esta idea es el método ρ debido a Pollard.

Supongamos que d un divisor propio desconocido de n y consideremos el epimorfismo canónico $\phi : \mathbb{Z}_n \to \mathbb{Z}_d$, dado por $\phi([x]_n) = [x]_d$, donde $[x]_n$ representa la clase de x módulo n. Consideremos una función polinómica $f : \mathbb{Z}_n \to \mathbb{Z}_n$ y partiendo de $a \in \mathbb{Z}_n$ elegido aleatoriamente, construimos de forma recursiva una sucesión (x_i) con

$$x_0 = a, \quad x_{i+1} = f(x_i).$$

Como cada clase módulo d es la unión de n/d clases módulo n, resulta más probable encontrar que dos de los x_i sean congruentes módulo d que que lo sean módulo n. Por tanto es razonable pensar que nuestra en en la lista generada aparezcan x_i y x_j con $x_i \not\equiv x_j \mod n$ y $x_i \equiv x_j \mod d$. En tal caso $d|\mathrm{mcd}(x_i - x_j, n) \neq n$, con lo que $\mathrm{mcd}(x_i - x_j, n)$ será un factor propio de n. Por tanto, el siguiente algoritmo, que podría no detenerse nunca, sirve para obtener factores propios de n. Es conveniente señalar que no conviene aplicar el siguiente algoritmo a un número sobre el que no tengamos la certeza de que es compuesto. Claramente sobre un primo no servirá de nada.

Algoritmo 18. Factorización ρ de Pollard. Primera versión

ENTRADA: n y a dos números naturales y una aplicación polinómica $f : \mathbb{Z} \to \mathbb{Z}$.
$B := (a)$; $x := a$; $d := 1$.
Mientras que $d = 1$
  $x := f(x)$;
  Recorriendo $b \in B$ y mientras que $d = 1$.
    $d := \mathrm{mcd}(x - b, n)$;
    Si $d = n$, entonces
      SALIDA: "El algoritmo fue incapaz de encontrar un divisor propio de n".
    Añadir x a la lista B.
SALIDA: d, un divisor propio de n.

Obsérvese que hemos introducido una parada en el algoritmo, para el caso en que $\mathrm{mcd}(x - b, d) = n$. En la notación utilizada en el algoritmo la lista B estará formada por $b_1, b_2, \ldots, x = b_j$ y tendremos un $b = b_i$ con $i < j$ tal que $b_j \equiv b_i \mod n$. El haber llegado a ese punto significa que $\mathrm{mcd}(b_k - b_l, n) = 1$ para todo $k < l < j$ y $b_j \equiv b_i \mod n$. Entonces $\mathrm{mcd}(b_j - b_k, n) = \mathrm{mcd}(b_i - b_k, n) = 1$, para todo $i < k < j$. Es decir, en este paso del primer bucle no vamos a descubrir ningún divisor propio. De hecho, el haber elegido f como una función polinómica implica que tampoco lo vamos a descubrir en ningún paso posterior. Eso es consecuencia del siguiente lema cuya demostración es obvia:

Lema 4.8 *Sean f un polinomio con coeficientes enteros, n un entero y (b_i) una sucesión en la que $b_{i+1} = f(b_i)$ para todo i. Si $b_i \equiv b_j \mod n$ entonces $b_{i+s} \equiv b_{j+s} \mod n$ para todo $s \geq 0$.*

En particular, como $b_j \equiv b_i \mod n$ se tiene que $b_{j+s} \equiv b_{i+s} \mod n$, para todo $s \geq 0$. Por tanto, $b_{i+s+t(j-i)} = b_{j+s+(t-1)(j-i)} \equiv b_{i+s+(t-1)(j-i)} \equiv \cdots \equiv b_{i+s} \mod n$. Luego, si s_0 y t_0 son los restos de dividir s y t entre $j-i$ tenemos $\mcd(b_{i+s} - b_{i+t}, n) = \mcd(b_{i+s_0} - b_{i+t_0}, n) = 1$ ó n.

El inconveniente del Algoritmo 18 es el gran número de datos que tenemos que almacenar y de cálculos que es necesario hacer ya que tiene dos bucles encadenados y el interior va creciendo en tamaño, con lo que el número de cálculos va creciendo muy deprisa. Una forma de evitar esto es fijar una sucesión creciente $j_0 = 1 < j_1 < j_2 < \dots$ y, para cada j, solo comparar x_j con $x_{j_{k+1}}$ donde $j_k < j \leq j_{k+1}$. Se podría argumentar que esto podría suponer que el algoritmo no detectara un divisor propio $d = \mcd(x_j - x_i, n)$ con i un elemento que no esté en la sucesión (j_k). Sin embargo, si se elige la sucesión cuidadosamente esto solo retrasará un poco el descubrimiento de d. En efecto, vamos a elegir $j_i = 2^i$, o sea

$$j_0 = 1, \quad j_{i+1} = 2j_i.$$

De esta forma, en cada paso calculamos $\mcd(x_{2^h} - x_j, n)$ con $2^h < j \leq 2^{h+1}$. La sucesión x_j se va calculando recursivamente como antes: $x_{j+1} = f(x_j)$. En este segundo método, no hace falta acumular los valores x_j, sino simplemente se va manteniendo un contador j y la lista creciente B del Algoritmo 18 se sustituye por un b en el que se guarda el último valor de la forma x_{2^h}. El valor de b se actualiza cada vez que el contador j toma un valor potencia de 2.

Algoritmo 19. Factorización ρ de Pollard. Segunda versión

ENTRADA: n y a dos números naturales y $f \in \mathbb{Z}[X]$.
$i := 1$; $j := 2$; $b := a$; $x := a$; $d := 1$;
Mientras que $d = 1$
  $x := f(x)$;
  $d := \mcd(x - b, n)$;
  Si $d = n$, entonces
    SALIDA: "El algoritmo fue incapaz de encontrar un divisor propio de n".
  $i := i + 1$;
  Si $i = j$, entonces
    $j := j^2$; $b := x$;
SALIDA: d, un divisor propio de n.

Análisis del tiempo de cálculo

Recordemos que el Algoritmo 18 tiene dos bucles: El exterior en el que se actualiza el valor de x y la lista B se aumenta con el valor de x. Por otro lado, el bucle interior en el que se recorren los valores de b de B. En cambio, el Algoritmo 19 tiene un único bloque que corresponde al

bloque exterior del Algoritmo 18. Podemos considerar que el bloque interior del Algoritmo 18 se sustituye en el Algoritmo 19 por una actualización del valor de b. La siguiente proposición justifica la afirmación anterior de que el descubrimiento del divisor propio con el Algoritmo 19 solo se retrasará un poco respecto del encuentro con el Algoritmo 18. La pérdida en el número de bucles mayor del Algoritmo 19 con respecto al Algoritmo 18 se compensa con creces, pues cada bucle exterior del Algoritmo 18, que es un bucle con un número de pasos creciente, se sustituye en el Algorimo 19 por un único paso de dicho bucle interior.

Proposición 4.9 *El número de bucles que completa el Algoritmo 19 antes de parar es a lo sumo 4 veces mayor que el número de bucles exteriores que completa el Algoritmo 18 con la misma semilla y función polinómica.*

Demostración. Obsérvese que en el Algoritmo 19, $i = 2^h < j \leq 2^{h+1}$. Es decir, i recorre las potencias de 2 y es el menor número con el mismo número de bits que j. Supongamos que el Algoritmo 18 se para con los valores $i_1 < i_2$ y el Algoritmo 19 se para con los valores $j_1 = 2^k < j_2$. Por tanto, el Algoritmo 18 se ha parado en el bucle exterior i_2 y el Algoritmo 19 lo ha hecho en el bucle j_2 y tenemos que demostrar $j_2 \leq 4i_2$. Supongamos que i_2 tiene exactamente h bits, con lo que $2^{h-1} \leq i_2 < 2^h$ y, por tanto, $i_2 - i_1 < 2^h - 1$. Luego $j = 2^h + i_2 - i_1$ está en el intervalo $(2^h, 2^{h+1}]$. El hecho de que el Algoritmo 18 se haya parado con $i_1 < i_2$ significa que $d = \mathrm{mcd}(x_{i_1} - x_{i_2}, n) \neq 1$. Luego $x_{i_1} \equiv x_{i_2} \mod d$ y, del Lema 4.8, se deduce que $x_{2^h} \equiv x_j \mod d$, con lo que $\mathrm{mcd}(x_{2^h} - x_j, n) \neq 1$. En consecuencia, el Algoritmo 19 se parará en el bucle j, si es que no lo ha hecho antes. Es decir $j_2 \leq j \leq 2^{h+1} = 4 \cdot 2^{h-1} < 4i_2$. ∎

Cabría cuestionar qué ventaja ofrece la Factorización ρ de Pollard sobre la búsqueda sistemática de divisores $2, 3, 5, \ldots$. Suponiendo que el número es impar, la búsqueda sistemática de divisores consiste en ir probando con todos los números impares, $3, 5, 7, \ldots$. Podemos plantear esto como un caso particular de la Factorización ρ de Pollard, en la que la semilla es 3 y la función polinómica es $f(x) = x + 2$. Si el menor divisor de n es d las diferencias entre los valores obtenidos de esta forma están todas en diferentes clases módulo d durante los primeros $(d-3)/2$ pasos. Es decir tenemos garantizado el fracaso durante un número de pasos que crece exponencialmente con respecto a la longitud de d. La idea de elegir una función f diferente es que la sucesión obtenida $f(x)$ tenga un comportamiento aleatorio con la esperanza de que el número pasos fallidos no sea tan grande.

Vamos a analizar la probabilidad de que si f tiene un comportamiento aleatorio y la semilla x también se elige de forma aleatoria entonces después de m pasos el algoritmo no haya parado. Supongamos que d es un divisor propio de n. El que el algoritmo no haya parado después de m pasos implica que todos los elementos $x, f(x), \ldots, f^m(x)$ son diferentes módulo d. Para entender esto mejor cambiamos $\mathbb{Z}_d$ por un conjunto arbitrario S de cardinal d y elegimos al azar una aplicación $f : S \to S$ una semilla $x \in S$. Es decir elegimos un elemento $(f, x) \in A = S^S \times S$ al azar. Ahora queremos calcular la probabilidad de que la sucesión $x, f(x), \ldots, f^m(x)$ no tenga elementos repetidos o lo que es lo mismo que el cardinal del siguiente conjunto sea exactamente $m + 1$:

$$X_{f,x}^m = \{x_1 = x, x_2 = f(x_1), \ldots, x_{m+1} = f(x_m)\}.$$

Pongamos pues

$$B_m = \{(f, x) \in A : |X_{f,x}^m| = m + 1\}.$$

Entonces la probabilidad que queremos calcular es precisamente $p_m = \frac{|B_m|}{|A|}$. Claramente, si $m \geq d$ entonces $B_m = \emptyset$, con lo que $p_m = 0$. Supongamos que $m < d$. Entonces existen $V_d^{m+1} = d(d-1)\dots(d-m)$ sucesiones $x_1, x_2, \dots, x_{m+1}$ de elementos distintos de S. Por tanto, existen $V_d^{m+1} d^{d-m}$ parejas $(f, x) \in A$ tales que $|X_{f,x}^m| = m+1$, o sea, $|B_m| = V_d^{m+1} d^{d-m}$. Luego

$$p_m = \frac{d^{d-m} \prod_{j=0}^{m}(d-j)}{d^{d+1}} = \frac{\prod_{j=0}^{m}(d-j)}{d^{m+1}} = \prod_{j=0}^{m} \frac{d-j}{d} = \prod_{j=0}^{m} \left(1 - \frac{j}{d}\right).$$

Utilizando que $\ln(x) < x - 1$ para $x \neq 1$, tenemos que

$$\ln p_m = \sum_{j=0}^{m} \ln\left(1 - \frac{j}{d}\right) \leq \sum_{j=0}^{m} -\frac{j}{d} = -\frac{m(m+1)}{2d} < -\frac{m^2}{2d}.$$

Esto demuestra el siguiente

Lema 4.10 *Sea S un conjunto finito con d elementos y sean $f : S \to S$ y $a \in S$ elegidos al azar con distribución uniforme. Entonces la probabilidad de que la sucesión $x, f(x), f^2(x) \dots, f^m(x)$ no tenga elementos repetidos es menor que $e^{-\frac{m^2}{2d}}$.*

La moraleja es que la probabilidad de que los algoritmos 18 y 19 requieran más de m pasos desciende exponencialmente con m. Con esto podemos estimar el tiempo de calculo esperable para estos algoritmos.

Proposición 4.11 *Sea n un entero impar compuesto. Sean f un polinomio con coeficientes enteros y $a \in \mathbb{Z}_n$. Supongamos que la sucesión (x_i) obtenida poniendo $x_1 = a$ y $x_{i+1} = f(x_i)$ tienen un comportamiento como el de una función media. Si despreciamos la posibilidad de que el método ρ se pare porque encuentre una diferencia múltiplo de n, entonces la segunda versión del método ρ encontrará un factor $d < \sqrt{n}$ en un tiempo $O(\sqrt[4]{n} \log^3 n)$, con una alta probabilidad. Más precisamente, existe una constante C tal que para cada número real $\lambda > 0$, la probabilidad de que el método ρ no encuentre el factor d en $C\sqrt{\lambda}\sqrt[4]{n} \log^3 n$ operaciones bit es menor que $e^{-\lambda}$.*

Demostración. Sabemos que existe una constante C_1 tal que para cada $x < n$ el tiempo de calculo de $\text{mcd}(x, n)$ es $\leq C_1 \log^3 n$ y el tiempo de cálculo de $(f(x) \mod n)$ es $C_1 \log^2 n$. De acuerdo con el Lema 4.10, la probabilidad de que el primer método no haya encontrado el factor en $m = 1 + \left\lfloor \sqrt{2\lambda d} \right\rfloor$ bucles exteriores es menor que $e^{-\lambda}$. De acuerdo con la comparación de los bucles exteriores de los Algoritmos 18 y 19 que hicimos anteriormente, la misma cota de probabilidad tendremos para el segundo método en el bucle en el que el mayor número es menor que $4\left(1 + \left\lfloor \sqrt{2\lambda d} \right\rfloor\right)$ bucles. Como cada bucle exterior del Algoritmo 18 tiene una longitud como la suma de los anteriores, el número de bucles interiores totales realizados es menor que $8(1 + \left\lfloor \sqrt{2\lambda d} \right\rfloor) \leq 8(1 + \sqrt{2}\sqrt{\lambda}\sqrt[4]{n})$, con lo que el tiempo total es menor que $8(1 + \sqrt{2}\sqrt{\lambda}\sqrt[4]{n})C_1(\log^3 n + \log^2 n)$. Cogiendo C_1 un poquito más grande para eliminar el molesto 1, tendremos que el número total de operaciones realizado es menor que $C\sqrt{\lambda}\sqrt[4]{n} \log^3 n$. ∎

De todas formas este tiempo "probabilístico" sigue siendo exponencial.

4.5 Bases de Factores

Los métodos de factorización que hemos visto hasta ahora pueden ser muy efectivos pero a menudo no consiguen producir divisores. En tal caso necesitaremos utilizar herramientas más sofisticadas que resultan más efectivas pero que a la vez son bastante más complejas. Es recomendable comenzar utilizando las herramientas más simples (divisores pequeños, Fermat, $p-1$ y ρ de Pollard) antes de probar con las herramientas más pesadas que vamos a explicar ahora. Por supuesto nunca se debe olvidar que antes de intentar factorizar un número hay que asegurarse de que no es primo utilizando los algoritmo de los vistos en el Capítulo 3. Los dos métodos sofisticados de factorización de los que estamos hablando son la Factorización con Fracciones Continuas o la Criba Cuadrática. Ambos tienen una característica común que vamos a introducir en esta sección y que está conectada de forma remota con la Factorización de Fermat.

En la Sección 4.3 hemos visto que el problema de factorizar un número impar es equivalente al de escribirlo como diferencia de cuadrados. En otras palabras, si $x^2 \equiv y^2 \mod n$, entonces n divide a $(x+y)(x-y)$ y cabe esperar que los divisores de n se repartan entre los divisores de $x+y$ y $x-y$ de forma que bien $\mcd(x+y,n)$ ó $\mcd(x-y,n)$ sea un divisor propio de n. En esta sección vamos a ver un método para encontrar un divisor propio que utiliza esta idea. Se trata de intentar buscar dos números x e y tales que $x^2 \equiv y^2 \mod n$, con la esperanza de que bien $\mcd(x+y,n)$ ó bien $\mcd(x-y,n)$ sea un divisor propio de n. Por tanto nuestro objetivo es encontrar números enteros x e y que satisfagan

$$x^2 \equiv y^2 \mod n \quad \text{y, o bien} \quad x \not\equiv y \mod n \quad \text{o bien} \quad x \not\equiv -y \mod n.$$

Vamos a denotar por "$(a \mod n)$" al entero congruente con a, módulo n que es menor en valor absoluto, con preferencia para el caso positivo en caso de ambigüedad. Es decir,

$$r = (a \mod n) \quad \Leftrightarrow \quad \begin{cases} a \equiv r \mod n, \\ -\frac{n}{2} < r \leq \frac{n}{2}. \end{cases}$$

Definición 4.12 *Una* base de factores *es una lista de primos distintos:*

$$B = (p_1, \ldots, p_k).$$

*Dada es una base de factores B, decimos que un entero b es un B-*número *módulo n si* ($b^2 \mod n$) *es un producto de elementos de B, o sea todos los primos que dividen a n están en B.*

Sea $B = (p_1, \ldots, p_k)$ una base de factores y pongamos $p_0 = -1$. Si b es un B-número módulo n entonces

$$(b^2 \mod n) = p_0^{\alpha_0} p_1^{\alpha_1} \cdots p_k^{\alpha_k},$$

para ciertos enteros $\alpha_0, \alpha_1, \ldots, \alpha_k$. En tal caso, vamos a utilizar la siguiente notación

$$\epsilon_i(b) = (\alpha_i \mod 2) \quad (i = 0, 1, \ldots, k)$$

y

$$\epsilon(b) = (\epsilon_0(b), \epsilon_1(b), \ldots, \epsilon_k(b)).$$

Ejemplo 4.13 Pongamos $n = 59051$ y sea B la base de factores formada por los primos menores o iguales que 100, o sea

$$B = (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97),$$

Si $b = 244$ entonces $(b^2 \mod n) = 485 = 5 \cdot 97$. Por tanto b es un B-número y, como 5 y 97 ocupan las posiciones 4 y 26 de B, tenemos $\epsilon_i(b) = 1$, si $i = 3$ ó $i = 25$ y $\epsilon_i(b) = 0$ para todos los demás valores de i. Es decir,

$$\epsilon(244) = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1).$$

Los números 245 y 246 no son B-números módulo n pues $(245^2 \mod n) = 974 = 2 \cdot 487$ y $(246^2 \mod n) = 1465 = 5 \cdot 293$. El siguiente B-número módulo n es 247 pues $(247^2 \mod n) = 1958 = 2 \cdot 11 \cdot 89$ y por tanto $\epsilon(247)$ tiene unos en las posiciones 2, 6 y 25 y ceros en las demás posiciones. Obsérvese que $(249^2 \mod n) = 2950 = 2 \cdot 5^2 \cdot 59$, con lo que 249 es un B-número módulo n y aunque 5 divide a $(249^2 \mod n) = 2950$, se tiene que $\epsilon_3(249) = 0$. Es decir

$$\epsilon(249) = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0).$$

¿ Sean $b_1, \ldots, b_m$, B-números módulo n tales que $\sum_{j=1}^{m} \epsilon(b_i) \equiv 0 \mod 2$ en $\mathbb{Z}_2^k$, es decir,

$$\sum_{j=1}^{m} \epsilon_i(b_j) \equiv 0 \mod 2 \tag{4.1}$$

para todo $i = 0, 1, \ldots, k$. Esto implica que si

$$a_j = (b_j^2 \mod n) = \prod_{i=1}^{k} p_i^{\alpha_{ji}},$$

entonces

$$\sum_{j=1}^{m} \alpha_{ji} \equiv \sum_{j=1}^{m} \epsilon_i(b_j) \equiv 0 \mod 2.$$

Pongamos

$$b = \prod_{j=1}^{m} b_j, \quad \gamma_i = \frac{\sum_{j=1}^{m} \alpha_{ji}}{2} \quad \text{y} \quad c = \prod_{i=1}^{k} p_i^{\gamma_i}.$$

Entonces

$$b^2 = \prod_{j=1}^{m} b_j^2 \equiv \prod_{i=1}^{k} p_i^{\sum_{j=1}^{m} \alpha_{ji}} \equiv \prod_{j=1}^{k} p_i^{2\gamma_i} = c^2 \mod n.$$

En resumen,

$$b^2 \equiv c^2 \mod n,$$

es decir, tendremos dos números b y c tales que $b^2 \equiv c^2 \mod n$. Eso implica que si $d_1 = \text{mcd}(b+c, n)$ y $d_2 = \text{mcd}(b-c, n)$, entonces d_1 y d_2 son dos divisores de n y $n | d_1 d_2$. Si además,

$$b \not\equiv c \mod n \quad \text{ó} \quad b \not\equiv -c \mod n,$$

entonces d_1 ó d_2 es un divisor propio de n.

Ejemplo 4.14 Pongamos $n = 154381$ y consideremos los siguientes valores $b_1 = 13167$, $b_2 = 22578$, $b_3 = 118799$. Calculando $(b_i^2 \mod n)$ para todos ellos obtenemos

$$\begin{array}{rcccl} (13167^2 \mod n) & = & 26 & = & 2 \cdot 13, \\ (22578^2 \mod n) & = & 22 & = & 2 \cdot 11, \\ (118799^2 \mod n) & = & 143 & = & 11 \cdot 13. \end{array}$$

Con lo que si elegimos como base de factores $B = \{2, 11, 13\}$ tenemos

$$\begin{array}{rcl} \epsilon(b_1) & = & (1, 0, 1) \\ \epsilon(b_2) & = & (1, 1, 0) \\ \epsilon(b_3) & = & (0, 1, 1) \end{array}$$

Luego $\epsilon(b_1) + \epsilon(b_2) + \epsilon(3) = (2, 2, 2) \equiv (0, 0, 0) \mod 2$. Por tanto, si ponemos $b = b_1 b_2 b_3 = 35317104404274 \equiv 10280 \mod n$ y $c = 2 \cdot 11 \cdot 13 = 286$ tenemos $b^2 \equiv c^2 \mod n$. Calculando $\mcd(b - c, n) = 263$ y $\mcd(b + c, n) = 587$, obtenemos la factorización $n = 263 \cdot 587$.

El lector se preguntará de dónde han salido los valores de b_i para obtener tan rápidamente un resultado favorable y sospechará, con razón, que el ejemplo ha sido diseñado para que el resultado saliera fácil. En la práctica nos encontramos con el problema de encontrar una familia $b_1, \ldots, b_m$ de B-números módulo n tales que $\sum_{i=1}^{m} \epsilon(b_i) \equiv 0 \mod 2$. Si elegimos los elementos b_i, de forma que $|b_i| < \sqrt{\frac{n}{2}}$, entonces $a_i = (b_i^2 \mod n) = b_i^2$, con lo que $b = c$ y no hemos sacado nada en claro.

¿Cuál es la probabilidad de que eligiendo los b_i al azar resulte que $b \not\equiv \pm c \mod n$. Obsérvese que c es una raíz cuadrada de $(b^2 \mod n)$. Si n tiene r factores primos impares diferentes, entonces del Teorema Chino de los Restos se deduce que cada cuadrado módulo n tiene 2^r raíces cuadradas. Luego la probabilidad de que $c = \pm(b \mod n)$ es $\frac{2}{2^r} = \frac{1}{2^{r-1}}$. Si $r \geq 2$, ésta probabilidad es $\leq \frac{1}{2}$.

En la práctica se suelen elegir la base de factores y los b_i de una de las formas siguientes. La primera es elegir B formado por los $k - 1$ primeros primos y después elegir los b_i al azar hasta tener suficientes para encontrar una relación como en (4.1). La segunda es elegir los b_i tal que los factores primos de $(b_i^2 \mod n)$ sean pequeños y elegir la base de factores B de forma que todos los b_i sean B-números módulo n, es decir B está formada por los primos que dividen a algún b_i. Una forma de conseguir esto es elegir los b_i cercanos a $\sqrt{kn}$ para k pequeño.

Ejemplo 4.15 Sea $n = 4633$. Entonces $\sqrt{4633} \approx 68$ y

$$\begin{array}{rcl} (67^2 \mod n) = (4489 \mod n) & = & -144 = -2^4 3^2 \\ (68^2 \mod n) = (4624 \mod n) & = & -9 = -3^2 \end{array}$$

Si ponemos $B = \{2, 3\}$, entonces $\epsilon(67) = (1, 0, 0)$ y $\epsilon(68) = (1, 0, 0)$. Sean $b = (67 \cdot 68 \mod n) = (4556 \mod n) = -77$ y $c = 2^2 3^2 = 36$. Entonces $77 \not\equiv 36 \mod n$ y $\mcd(-77 + 36, 4633) = \mcd(-41, 4633) = 41$.

Se puede estimar heurísticamente el tiempo necesario para encontrar un factor con este método como $O(e^{C\sqrt{\log n \log\log n}})$[1]. En las dos siguientes secciones vamos a ver variaciones del método introducido en ésta.

[1] N. Koblitz, A course in number theory and cryptography, Springer-Verlag, 1988.

4.6 Fracciones continuas

En la sección anterior se ha visto la necesidad de encontrar números b tales que $(b^2 \mod n)$ sea pequeño para poder utilizarlos como bases de factores con la intención de factorizar n. En esta sección veremos como obtener esto utilizando fracciones continuas.

Una fracción continua es una expresión del siguiente tipo

$$\alpha = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \ldots}},$$

para $a_0, a_1, a_2, \ldots$ números reales positivos. La fracción continua puede tener una cantidad finita o infinita de términos. De forma más rigurosa: Dada una sucesión finita de números reales positivos $a_1, \ldots, a_n$ definimos la fracción continua finita $[a_1, \ldots, a_n]$ de forma recursiva poniendo

$$[a] = a, \quad [a_1, \ldots, a_n] = a_1 + \frac{1}{[a_2, \ldots, a_n]}.$$

Lema 4.16 *Si $1 \leq k < n$ entonces*

$$[a_1, \ldots, a_n] = [a_1, \ldots, a_k, [a_{k+1}, \ldots, a_n]].$$

Demostración. Por inducción en k, con el caso $k = 1$ obvio. Si se verifica para $k < n$, entonces

$$\begin{aligned} [a_1, \ldots, a_n] &= [a_1, \ldots, a_k, [a_{k+1}, \ldots, a_n]] = a_1 + \frac{1}{[a_2, \ldots, a_k, [a_{k+1}, \ldots, a_n]]} \\ &= a_1 + \frac{1}{[a_2, \ldots, a_k, a_{k+1}, [a_{k+2}, \ldots, a_n]]} = [a_1, \ldots, a_{k+1}, [a_{k+2}, \ldots, a_n]]. \end{aligned}$$

∎

Sea α un número real positivo. Construimos dos sucesiones (a_i) y (x_i) de forma recursiva:

$$\begin{aligned} a_0 &= \lfloor \alpha \rfloor, & x_0 &= \alpha - a_0, \\ a_i &= \left\lfloor \frac{1}{x_{i-1}} \right\rfloor, & x_i &= \frac{1}{x_{i-1}} - a_i \quad (i \geq 1). \end{aligned}$$

Si $x_i = 0$ entonces no podremos construir a_{i+1}. Esto pasará cuando en el paso anterior, x_{i-1} sea el inverso de un entero. En tal caso la sucesión resultante será finita: $[a_0, \ldots, a_i]$.

Ejemplo 4.17 Comenzamos calculando la fracción continua de $\frac{123}{190}$:

$$\begin{aligned} a_0 &= \lfloor \alpha \rfloor = 0, & x_0 &= \tfrac{123}{190}; \\ a_1 &= \left\lfloor \tfrac{190}{123} \right\rfloor = 1, & x_1 &= \tfrac{190}{123} - 1 = \tfrac{67}{123}; \\ a_2 &= \left\lfloor \tfrac{123}{67} \right\rfloor = 1, & x_3 &:= \tfrac{123}{67} - 1 = \tfrac{56}{67}; \\ a_4 &= \left\lfloor \tfrac{67}{56} \right\rfloor = 1, & x_4 &= \tfrac{67}{56} - 1 = \tfrac{11}{56}; \\ a_4 &= \left\lfloor \tfrac{56}{11} \right\rfloor = 5, & x_4 &= \tfrac{56}{11} - 5 = \tfrac{1}{11}; \\ a_5 &= 11, & x_5 &= 0. \end{aligned}$$

Por tanto la fracción continua de $\frac{123}{190}$ es $[0,1,1,1,5,11]$.

Consideremos ahora la fracción continua de $\sqrt{2}$:

$$a_0 = \lfloor\sqrt{2}\rfloor = 1, \qquad x_0 = \sqrt{2}-1;$$

$$a_1 = \left\lfloor \tfrac{1}{\sqrt{2}-1}\right\rfloor = 2, \quad x_1 = \tfrac{1}{\sqrt{2}-1} - 2 = \tfrac{3-2\sqrt{2}}{\sqrt{2}-1} = (3-2\sqrt{2})(\sqrt{2}+1) = \sqrt{2}-1 = x_0.$$

Está claro que en cuanto dos x_i consecutivos se repitan a partir de aquí la sucesión de los a_i será constante. Por tanto la fracción continua de $\sqrt{2}$ es $[1,2,2,\dots]$.

Finalmente para $1+\sqrt{7}$ obtenemos:

$$a_0 = 3, \qquad\qquad x_0 = \alpha - 3 = \sqrt{7}-2;$$

$$a_1 = \left\lfloor \tfrac{1}{\sqrt{7}-2}\right\rfloor = \left\lfloor \tfrac{\sqrt{7}+2}{3}\right\rfloor = 1, \qquad x_1 = \tfrac{\sqrt{7}+2}{3} - 1 = \tfrac{\sqrt{7}-1}{3};$$

$$a_2 = \left\lfloor \tfrac{3}{\sqrt{7}-1}\right\rfloor = \left\lfloor \tfrac{\sqrt{7}+1}{2}\right\rfloor, \qquad x_2 = \tfrac{\sqrt{7}+1}{2} - 1 = \tfrac{\sqrt{7}-1}{2};$$

$$a_2 = \left\lfloor \tfrac{2}{\sqrt{7}-1}\right\rfloor = \left\lfloor \tfrac{\sqrt{7}+1}{3}\right\rfloor = 1, \qquad x_3 = \tfrac{\sqrt{7}+1}{3} - 1 = \tfrac{\sqrt{7}-2}{3};$$

$$a_4 = \left\lfloor \tfrac{3}{\sqrt{7}-2}\right\rfloor = \lfloor\sqrt{7}+2\rfloor = 4, \quad x_4 = \sqrt{7}-2;$$

$$a_4 = \left\lfloor \tfrac{1}{\sqrt{7}-2}\right\rfloor = \left\lfloor \tfrac{\sqrt{7}+2}{3}\right\rfloor = 1, \qquad x_4 = \tfrac{\sqrt{7}+2}{3} - 1 = x_1.$$

Por tanto, a partir de aquí la fracción continua se comienza a repetir y la fracción continua infinita será de la forma $[3,1,1,1,4,1,1,1,4\dots]$.

Utilizamos las fracciones continuas obtenidas para construir números reales. Para evitar tener que escribir cosas como la siguiente:

$$3+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{4+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{4+\dots}}}}}}}$$

vamos a utilizar la siguiente notación

$$\frac{1}{x+}y = \frac{1}{x+y}.$$

Así una fracción continua $[a_0,a_1,a_2,\dots]$ representa el número

$$a_0 + \frac{1}{a_1+}\frac{1}{a_2+}\dots.$$

Por ejemplo las tres fracciones continuas obtenidas representan los números:

$$\alpha = \frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{1}{5+}\frac{1}{11}, \quad \beta = 1+\frac{1}{2+}\frac{1}{2+}\dots, \quad \gamma = 3+\frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{1}{4+}\frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{1}{4+}\dots.$$

Entonces

$$\alpha = \frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{1}{5+}\frac{1}{11} = \frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{11}{56} = \frac{1}{1+}\frac{1}{1+}\frac{56}{67} == \frac{1}{1+}\frac{67}{123} = \frac{123}{190}$$

Si ponemos $\delta = \beta - 1 = \frac{1}{2+}\frac{1}{2} + \dots$ tenemos $\delta = \frac{1}{2+\delta}$, con lo que $\delta^2 + 2\delta - 1 = 0$, o sea $\delta = \frac{-2+2\sqrt{2}}{2} = -1 + \sqrt{2}$ y por tanto $\beta = \delta + 1 = \sqrt{2}$. Finalmente, si ponemos $\epsilon = \gamma - 3$, entonces

$$\epsilon = \frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{1}{4+}\frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{1}{4+}\cdots = \frac{1}{1+}\frac{1}{1+}\frac{1}{1+}\frac{1}{4+}\epsilon = \frac{1}{1+}\frac{1}{1+}\frac{4+\epsilon}{5+\epsilon} = \frac{1}{1+}\frac{5+\epsilon}{9+2\epsilon} = \frac{9+2\epsilon}{14+3\epsilon}$$

Luego $3\epsilon^2 + 12\epsilon - 9 = 0$ y por tanto $\epsilon = \frac{-12+\sqrt{12^2+9\cdot 12}}{6} = \frac{-12+6\sqrt{7}}{6} = -2 + \sqrt{7}$, con lo que $\gamma = 3 + \epsilon = 1 + \sqrt{7}$.

Hemos observado en los tres ejemplos que el número representado por la fracción continua es precisamente el número utilizado para construir la fracción continua. Esto es general como pronto veremos.

Obsérvese que los términos a_i de la fracción continua de un número real positivo α son números naturales y los términos x_i están en el intervalo $[0, 1)$, de donde se deduce que si $i > 0$, entonces $a_i > 1$. Además

$$x_i = \frac{1}{a_{i+1} + x_{i+1}}.$$

Utilizando esto y el Lema 4.16, es fácil ver por inducción sobre n que

$$\alpha = [a_0, a_1, \dots, a_{n-1}, a_n + x_n].$$

En efecto, $\alpha = a_0 + x_0 = [a_0 + x_0]$, por definición. Supongamos que $\alpha = [a_0, a_1, \dots, a_n + x_n]$. Entonces

$$\alpha = [a_0, a_1, \dots, a_n + \frac{1}{a_{n+1} + x_{n+1}}] = [a_0, a_1, \dots, [a_n, a_{n+1} + x_{n+1}]] = [a_0, a_1, \dots, a_n, a_{n+1} + x_{n+1}].$$

A partir de una fracción continua $[a_0, a_1, a_2, \dots]$ definimos dos sucesiones de números enteros de forma recursiva:

$$\begin{array}{ll} P_{-1} = 1, & Q_{-1} = 0, \\ P_0 = a_0, & Q_0 = 1 \\ P_n = a_n P_{n-1} + P_{n-2}, & Q_n = a_n Q_{n-1} + Q_{n-2}. \end{array}$$

Obsérvese que las fórmulas recursivas se pueden expresar matricialmente como

$$\begin{pmatrix} P_n & Q_n \\ P_{n-1} & Q_{n-1} \end{pmatrix} = \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} P_{n-1} & Q_{n-1} \\ P_{n-2} & Q_{n-2} \end{pmatrix}.$$

Ejemplo 4.18 Las fracciones continuas obtenidas en el Ejemplo 4.17 dan lugar a los siguientes primeros términos de las correspondientes sucesiones.

Para $\frac{123}{190}$ obtuvimos la fracción continua finita [0,1,1,1,5,11] que da lugar a

a_i		0	1	1	1	5	11
P_i	1	0	1	1	2	11	123
Q_i	0	1	1	2	3	17	190

Para $\sqrt{2}$, la fracción continua $[1, 2, 2, \dots]$ proporciona

a_i		1	2	2	2	2	2
P_i	1	1	3	7	17	41	99
Q_i	0	1	2	5	12	29	70

Finalmente para $1 + \sqrt{7}$ obtuvimos la fracción continua $[3, 1, 1, 1, 4, 1, 1, 1, 4 \dots]$ con la que obtenemos

a_i		3	1	1	1	4	1
P_i	1	3	4	7	11	51	62
Q_i	0	1	1	2	3	14	17

Lema 4.19 *Para todo $n \geq 1$ se tiene*

(1) $P_nQ_{n-1} - Q_nP_{n-1} = (-1)^{n+1}$.

(2) $\frac{P_n}{Q_n} - \frac{P_{n-1}}{Q_{n-1}} = \frac{(-1)^{n+1}}{Q_nQ_{n-1}}$.

(3) $P_nQ_{n-2} - Q_nP_{n-2} = (-1)^n a_n$.

(4) La sucesión $\left(\frac{P_{2n}}{Q_{2n}}\right)$ *es estrictamente creciente y la sucesión* $\left(\frac{P_{2n+1}}{Q_{2n+1}}\right)$ *es estrictamente decreciente.*

(5) La sucesión $(\frac{P_n}{Q_n})$ *es de Cauchy.*

(6) $\frac{P_n}{Q_n} = [a_0, a_1, \dots, a_n]$ *y esta fracción es reducida.*

(7) α *es racional si y solo si la sucesión* (a_i) *es finita (es decir, si* $x_i = 0$ *para algún* i*).*

Demostración. (1) se obtiene simplemente tomando determinantes en la ecuación

$$\begin{pmatrix} P_n & Q_n \\ P_{n-1} & Q_{n-1} \end{pmatrix} = \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_{n-1} & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix}$$

y (2) es consecuencia inmediata de (1).

(3)

$$\begin{aligned} P_nQ_{n-2} - Q_nP_{n-2} &= \begin{vmatrix} P_n & Q_n \\ P_{n-2} & Q_{n-2} \end{vmatrix} = \begin{vmatrix} a_nP_{n-1} + P_{n-2} & a_nQ_{n-1} + Q_{n-2} \\ P_{n-2} & Q_{n-2} \end{vmatrix} \\ &= a_n \begin{vmatrix} P_{n-1} & Q_{n-1} \\ P_{n-2} & Q_{n-2} \end{vmatrix} = (-1)^n a_n. \end{aligned}$$

(4) es consecuencia inmediata de (3) y de que todos los a_i con $i \geq 1$ son positivos.

Para demostrar (5) obsérvese que por (2), es suficiente demostrar que la sucesión (Q_n) es estrictamente creciente, y por tanto también lo es la sucesión (Q_nQ_{n-1}). Esto es consecuencia de la fórmula recursiva que define los Q_i, o sea $Q_n = a_nQ_{n-1} + Q_{n-2}$, y de que a_n es positivo para todo $n \geq 1$.

El hecho de que la fracción $\frac{P_n}{Q_n}$ sea reducida es consecuencia de (1). Razonamos por inducción sobre n para demostrar $\frac{P_n}{Q_n} = [a_0, a_1, \ldots, a_n]$, con el caso $n = 0$ trivial. Por hipótesis de inducción, si

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_{n-1} & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix}$$

Entonces

$$\frac{A}{B} = [a_1, \ldots, a_n]$$

Por tanto, como

$$\begin{pmatrix} P_n & Q_n \\ P_{n-1} & Q_{n-1} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix}$$

se tiene que

$$\frac{P_n}{Q_n} = \frac{Aa_0 + B}{A} = a_0 + \frac{1}{\frac{A}{B}} = a_0 + \frac{1}{[a_1, \ldots, a_n]} = [a_0, a_1, \ldots, a_n].$$

(7) Si α es irracional, entonces x_i también es irracional para todo i y por tanto la sucesión (a_i) es infinita. Supongamos que α es racional y pongamos $\alpha = \frac{a}{b}$ con a y b enteros. Entonces $a_0 = \left\lfloor \frac{a}{b} \right\rfloor$ es el cociente de la división entera de a entre b y $x_0 = \frac{a}{b} - a_0 = \frac{a - a_0 b}{b} = \frac{r_0}{b}$, es decir $r_0 = x_0 b$ es el resto de la división entera de a entre b. Entonces $a_1 = \left\lfloor \frac{b}{r_0} \right\rfloor$ es el cociente de la división entera de b entre r_0 y $x_2 = \frac{b}{r_0} - a_1 = \frac{b - r_0 a_1}{r_0} = \frac{r_1}{r_0}$ es el resto de dicha división. Más generalmente los a_i y $r_i = r_{i-1}x_{i-2}$ son los cocientes y restos que aparecen en el Algoritmo de Euclides para el cálculo del máximo común divisor de a y b. Como r_i es eventualmente 0, x_i es eventualmente 0. ∎

Lema 4.20 $\alpha = \frac{P_n + x_n P_{n-1}}{Q_n + x_n Q_{n-1}}$.

Demostración. Por inducción sobre n. Para $n = 0$

$$\frac{P_0 + x_0 P_{-1}}{Q_0 + x_0 Q_{-1}} = \frac{a_0 + x_0 \cdot 1}{1 + x_0 0} = a_0 + x_0 = \alpha.$$

Para $n > 0$

$$\begin{aligned} \alpha &= \frac{P_{n-1} + x_{n-1} P_{n-2}}{Q_{n-1} + x_{n-1} Q_{n-2}} = \frac{P_{n-1} + \frac{1}{a_n + x_n}(P_n - a_n P_{n-1})}{Q_{n-1} + \frac{1}{a_n + x_n}(Q_n - a_n Q_{n-1})} \\ &= \frac{(a_n + x_n)P_{n-1} + (P_n - a_n P_{n-1})}{(a_n + x_n)Q_{n-1} + (Q_n - a_n Q_{n-1})} = \frac{P_n + x_n P_{n-1}}{Q_n + x_n Q_{n-1}}. \end{aligned}$$

∎

De los Lemas 4.19 y 4.20 se deduce que

$$\alpha - \frac{P_n}{Q_n} = \frac{P_n + x_n P_{n-1}}{Q_n + x_n Q_{n-1}} - \frac{P_n}{Q_n} = \frac{(-1)^n x_n}{(Q_n + Q_{n-1}x_n)Q_n} \tag{4.2}$$

y, por tanto, mientras que $x_n \neq 0$, las fracciones $\frac{P_n}{Q_n} = [a_0, a_1, \ldots, a_n]$ son alternativamente mayor y menor que α. Combinando esto con el apartado (4) del Lema 4.19 tenemos

$$\frac{P_2}{Q_2} < \frac{P_4}{Q_4} < \ldots \frac{P_{2(n-1)}}{Q_{2(n-1)}} < \frac{P_{2n}}{Q_{2n}} < \cdots < \alpha < \cdots < \frac{P_{2n+1}}{Q_{2n+1}} < \frac{P_{2n-1}}{Q_{2n-1}} < \cdots < \frac{P_3}{Q_3} < \frac{P_1}{Q_1}. \tag{4.3}$$

Además, como la sucesión $\left(\frac{P_n}{Q_n}\right)$ es de Cauchy, se tiene:

Teorema 4.21 $\alpha = \lim_{n\to\infty} \frac{P_n}{Q_n} = \lim_{n\to\infty}[a_0, a_1, \ldots, a_n]$.

A la vista del Teorema anterior no resulta extraño que las fracciones $\frac{P_n}{Q_n} = [a_0, a_1, \ldots, a_n]$ se llamen *convergentes de la fracción continua* de α. En el caso en que α sea racional, el último convergente es igual a α y, en el caso en que α sea irracional, la sucesión de convergentes converge a α, tomando alternadamente valores mayores y menores que α.

Como $\text{mcd}(P_i, Q_i) = 1$, la fracción $\frac{P_i}{Q_i}$ es reducida. Nuestro interés por los convergentes de una fracción continua radica en que los numeradores de los convergentes de $\sqrt{n}$ resultan útiles en la factorización de n utilizando bases de factores. Esto está basado en la siguiente proposición.

Proposición 4.22 *Si $\alpha > 1$, entonces, para todo $n \geq 1$ se tiene*

$$|P_n^2 - \alpha^2 Q_n^2| < 2\alpha.$$

Demostración. Como α está entre $\frac{P_n}{Q_n}$ y $\frac{P_{n+1}}{Q_{n+1}}$, aplicando el Lema 4.19 tenemos

$$\left|\alpha - \frac{P_n}{Q_n}\right| < \frac{1}{Q_n Q_{n+1}}$$

y

$$\left|\alpha + \frac{P_n}{Q_n}\right| = \left|2\alpha + \frac{P_n}{Q_n} - \alpha\right| \leq 2\alpha + \left|\frac{P_n}{Q_n} - \alpha\right| \leq 2\alpha + \frac{1}{Q_n Q_{n+1}}.$$

Entonces

$$\begin{aligned} |P_n^2 - \alpha^2 Q_n^2| &= Q_n^2 \left|\alpha^2 - \frac{P_n^2}{Q_n^2}\right| = Q_n^2 \left|\alpha - \frac{P_n}{Q_n}\right| \left|\alpha + \frac{P_n}{Q_n}\right| \\ &< Q_n^2 \frac{1}{Q_n Q_{n+1}} \left(2\alpha + \frac{1}{Q_n Q_{n+1}}\right) = 2\alpha \left(\frac{Q_n}{Q_{n+1}} + \frac{1}{2\alpha Q_{n+1}^2}\right) \end{aligned}$$

y, por tanto,

$$\begin{aligned} |P_n^2 - \alpha^2 Q_n^2| - 2\alpha &< 2\alpha\left(\tfrac{Q_n}{Q_{n+1}} + \tfrac{1}{2\alpha Q_{n+1}^2} - 1\right) < 2\alpha\left(\tfrac{Q_n}{Q_{n+1}} + \tfrac{1}{Q_{n+1}^2} - 1\right) \\ &< 2\alpha\left(\tfrac{Q_{n+1}^2}{Q_{n+1}^2} - 1\right) = 0. \end{aligned}$$

∎

Corolario 4.23 *Si* $\alpha = \sqrt{n} \notin \mathbb{Z}$, *entonces* $|(P_i^2 \mod n)| < 2\sqrt{n}$.

Demostración. De la Proposición 4.22 tenemos

$$|(P_i^2 \mod n)| = |(P_i^2 - nQ_i^2 \mod n)| \leq |P_i^2 - nQ_i^2| < 2\sqrt{n}.$$

∎

Del Corolario 4.23 se tiene que los numeradores P_i de las fracciones reducidas de los $[a_0, \ldots, a_n]$ son buenos candidatos para producir una base de factores B en la que estos números sean B-números, ya que sus cuadrados módulo n son menores que $2\sqrt{n}$.

He aquí el algoritmo de factorización con fracciones continuas que no es más que el de bases de factores con la variación de que las bases de factores las construimos con la fracción continua de la raíz cuadrada del número que queremos factorizar.

Algoritmo 20. Algoritmo de Factorización con Fracciones Continuas

ENTRADA: n, un entero positivo.
INICIALIZACIÓN:
$B := (); P_{-1} = 1; P_0 := \lfloor \sqrt{n} \rfloor ; a_0 := \lfloor \sqrt{n} \rfloor ; x_0 := \sqrt{n} - a_0; i := 0, D := \emptyset, j := 0.$
Mientras que $D = \emptyset$.

- $t_i := (P_i^2 \mod n)$.
- Se intenta descomponer t_i en producto de factores primos. Si no se consigue se desecha t_i y se pasa al paso siguiente. Si se consigue:
  - $j := j + 1$; $s_j = t_i$, $Q_j = P_i$.
  - Se añaden los divisores primos de t_i que no aparezcan en B al final de la lista B.
  - Si $B = (p_1, \ldots, p_k)$ y $s_j = (-1)^{\alpha_0} p_1^{\alpha_1} \ldots p_k^{\alpha_k}$, entonces ponemos
    $$\epsilon_j := ((\alpha_0 \mod 2), (\alpha_1 \mod 2), \ldots, (\alpha_k \mod 2))$$
    y añadimos ceros a los ϵ_j, con $j < i$ hasta que ϵ_j tenga longitud $k + 1$.
  - Se busca una relación de dependencia lineal módulo 2 entre los ϵ_j. Si no se encuentra se pasa al paso siguiente. Si se consigue y la relación de dependencia es
    $$\sum_{j \in J} \epsilon_j \equiv 0 \mod 2,$$
    ponemos
    $$b := \prod_{j \in J} Q_j, \quad c := \sqrt{\prod_{j \in J} s_j}.$$
  - $D := \{d \in \{\text{mcd}(b + c, n), \text{mcd}(b - c, n)\} : d \neq 1, n\}$.
  - Si $D \neq \emptyset$ entonces SALIDA: D.
- $i = i + 1, a_i = \left\lfloor \frac{1}{x_{i-1}} \right\rfloor , x_i = \frac{1}{x_{i-1}} - a_i, P_i = a_i P_{i-1} + P_{i-2}$.

Ejemplo 4.24 Vamos a aplicar el Algoritmo 20 al número $n = 305172473$. Para simplificar, en lugar de poner una base de factores dinámica, vamos a considerar la base de factores formada por todos los primos menores que 100. En la siguiente tabla incluimos: Los primeros valores de a_i, o sea los primeros términos de la expansión de $\sqrt{n}$ como fracción continua; los correspondientes numeradores P_i de los convergentes; así como $t_i = (P_i^2 \mod n)$ y la

factorización de t_i.

i	a_i	P_i	$t_i = (P_i^2 \mod n)$
-1		1	
0	17469	17469	$-6512 = -2^4 \cdot 11 \cdot 37$
1	5	87344	-337489
2	2	192157	$-1556584 = -2^3 \cdot 179 \cdot 1087$
3	1	279501	$-3344087 = -17 \cdot 229 \cdot 859$
4	2	751159	$-24059296 = -2^4 \cdot 751853$

Como se ve el único de los primeros valores de t_i que es producto de los elementos de la base de factores elegida es el primero. Los dos siguientes P_i para los que el correspondiente t_i es producto de los elementos de la base elegida son

P_i	t_i
236162819	$-61505024 = 2^9 \cdot 7 \cdot 131^2$
122252236	$47038377 = 3 \cdot 29 \cdot 31 \cdot 107 \cdot 163$

Por tanto los tres primeros vectores ϵ son los siguientes:

$$(1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$
$$(1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$
$$(0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0)$$

Está claro que no hay ninguna relación de dependencia lineal entre estos tres vectores. Sería muy largo escribir todos los cálculos que el programa realiza para completar el proceso. Básicamente sigue calculando los numeradores P_i de los convergentes utilizando la regla de construcción e intentando escribir $t_i = (P_i^2 \mod n)$ como producto de los primos de la base de factores. En cada paso va calculando el correspondiente vector $\epsilon(t_i)$ de ceros y unos y va construyendo una matriz cuyas filas son estos vectores hasta que el rango de esta matriz (sobre el cuerpo $\mathbb{F}_2$) sea menor que el número de filas. Al llegar a este momento se habrá encontrado una relación de dependencia lineal entre los ϵ_i. Los coeficientes de esta relación de dependencia serán ceros y unos. Se calculará b multiplicando los P_i que corresponden con un coeficiente 1 y c^2 multiplicando los correspondientes t_i. En realidad hay que guardar también los exponentes que aparecen en la factorización de los t_i con el fin de usarlos al calcular c. En nuestro caso los primeros exponentes son los siguientes:

$$(1,4,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$
$$(1,9,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$
$$(0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0)$$

Estos exponentes se ponen en otra matriz. Para calcular c se suman los vectores de esta matriz que corresponden a los coeficientes iguales a 1 de la relación de dependencia lineal. El resultado será un vector cuyas entradas serán todas múltiplos de 2 que dividiremos por 2 para obtener un vector $v = (v_i)$. Si $B = \{p_0, p_1, \ldots, p_k\}$ es la base de factores entonces $c = \prod_{i=1}^{k} p_i^{v_i}$.

En nuestro cálculo el programa realizó 1565 pasos antes de encontrar una relación de dependencia lineal entre los $\epsilon(P_i)$. De los t_i's obtenidos solo consiguió factorizar 31 que corresponden a los $A_i = P_{j_i}$ de la siguiente tabla:

i	A_i	$t_i = (A_i^2 \mod n)$
1	17469	$-6512 = -2^4 \cdot 11 \cdot 37$
2	236162819	$-61505024 = -2^9 \cdot 7 \cdot 131^2$
3	122252236	$47038377 = 3 \cdot 29 \cdot 31 \cdot 107 \cdot 163$
4	122625801	$20768748 = 2^2 \cdot 3 \cdot 7^2 \cdot 11 \cdot 13^2 \cdot 19$
5	157610438	$-121857184 = -2^5 \cdot 19 \cdot 43 \cdot 59 \cdot 79$
6	278727565	$-79998336 = -2^7 \cdot 3^2 \cdot 11 \cdot 59 \cdot 107$
7	113859236	$-2988700 = -2^2 \cdot 5^2 \cdot 11^2 \cdot 13 \cdot 19$
8	75167877	$-11793055 = -5 \cdot 83 \cdot 157 \cdot 181$
9	298808203	$-83846025 = -3^2 \cdot 5^2 \cdot 41 \cdot 61 \cdot 149$
10	36924630	$87840610 = 2 \cdot 5 \cdot 11 \cdot 13 \cdot 19 \cdot 53 \cdot 61$
11	53120187	$-116744056 = -2^3 \cdot 11 \cdot 13 \cdot 19 \cdot 41 \cdot 131$
12	286368086	$100118304 = 2^5 \cdot 3^2 \cdot 11^2 \cdot 13^2 \cdot 17$
13	46962364	$-129899070 = -2 \cdot 3^2 \cdot 5 \cdot 7 \cdot 41 \cdot 47 \cdot 107$
14	170839646	$-118767355 = -5 \cdot 7 \cdot 17 \cdot 31 \cdot 47 \cdot 137$
15	31384177	$131769300 = 2^2 \cdot 3 \cdot 5^2 \cdot 13^2 \cdot 23 \cdot 113$
16	290190550	$816753 = 3 \cdot 7 \cdot 19 \cdot 23 \cdot 89$
17	155912237	$6059105 = 5 \cdot 13 \cdot 31^2 \cdot 97$
18	205547858	$-1331121 = -3 \cdot 11^2 \cdot 19 \cdot 193$
19	72643746	$19437834 = 2 \cdot 3 \cdot 13 \cdot 17 \cdot 107 \cdot 137$
20	6677863	$-83715302 = -2 \cdot 11^2 \cdot 53 \cdot 61 \cdot 107$
21	304963704	$-55168278 = -2 \cdot 3 \cdot 11 \cdot 61 \cdot 71 \cdot 193$
22	90870556	$-143945900 = -2^2 \cdot 5^2 \cdot 7 \cdot 19 \cdot 79 \cdot 137$
23	184004209	$10969894 = 2 \cdot 13 \cdot 47^2 \cdot 191$
24	273224444	$60686244 = 2^2 \cdot 3^2 \cdot 43 \cdot 197 \cdot 199$
25	205370707	$-15019908 = -2^2 \cdot 3 \cdot 17^2 \cdot 61 \cdot 71$
26	103842717	$-47451820 = -2^2 \cdot 5 \cdot 13^2 \cdot 101 \cdot 139$
27	14968927	$-145526772 = -2^2 \cdot 3 \cdot 31 \cdot 37 \cdot 97 \cdot 109$
28	41253582	$-48900633 = -3 \cdot 47^3 \cdot 157$
29	210439856	$-139854330 = -2 \cdot 3^3 \cdot 5 \cdot 7^2 \cdot 11 \cdot 31^2$
30	93943918	$-17530616 = -2^3 \cdot 19 \cdot 29 \cdot 41 \cdot 97$
31	18665627	$-15601835 = -5 \cdot 17 \cdot 31^2 \cdot 191$

La relación de dependencia lineal se obtiene sumando los ϵ_i con $i = 7, 12, 18, 21, 23, 25, 29, 31$. Multiplicando los correspondientes P_i tenemos

$$b = (A_7 A_{12} A_{18} A_{21} A_{23} A_{25} A_{29} A_{31} \mod n) = 15890236$$

y multiplicando los t_i correspondientes y calculando su raíz cuadrada tenemos

$$c = (\sqrt{t_{12} t_{18} t_{21} t_{23} t_{25} t_{29} t_{31}} \mod n) = 44388428.$$

Calculando ahora $\mathrm{mcd}(n, b + c) = 23473$ y $\mathrm{mcd}(n, b - c) = 13001$, obtenemos dos divisores propios de n. De hecho estos dos números son primos y su producto es n.

En realidad el ejemplo anterior corresponde a un número demasiado pequeño para que mereciera la pena utilizar este método. Es importante, destacar que la fracción continua asociada a un número real que sea raíz de un polinomio de segundo grado es periódica. En particular, la fracción continua asociada a $\sqrt{n}$ es periódica y eso puede ser utilizado en la implementación del método. Sin embargo puede que el periodo sea muy largo.

Ataque a RSA utilizando fracciones continuas

Vamos a ver ahora una aplicación de las fracciones continuas en el criptoanálisis de RSA. Para eso analizamos cómo se aproximan los convergentes $\frac{P_n}{Q_n}$ de un número real positivo α o más concretamente las números $Q_n\alpha - P_n$. Del Lema 4.19 y de (4.3) se tiene

$$|Q_n\alpha - P_n| = Q_n\left|\alpha - \frac{P_n}{Q_n}\right| < Q_n\left|\frac{P_{n+1}}{Q_{n+1}} - \frac{P_n}{Q_n}\right| = \frac{1}{Q_{n+1}}$$

y

$$|Q_n\alpha - P_n| = Q_n\left|\alpha - \frac{P_n}{Q_n}\right| > Q_n\left|\frac{P_{n+2}}{Q_{n+2}} - \frac{P_n}{Q_n}\right| = \frac{a_{n+2}}{Q_{n+2}} \geq \frac{1}{Q_{n+2}}.$$

En resumen

$$\frac{1}{Q_{n+1}} < |Q_n\alpha - P_n| < \frac{1}{Q_{n+2}}. \tag{4.4}$$

y, en particular, la sucesión $(|Q_n\alpha - P_n|)$ es estrictamente decreciente, lo que en particular implica que la sucesión $\left|\alpha - \frac{P_n}{Q_n}\right|$ también es estrictamente decreciente. En realidad, esto es un caso particular de un resultado más general que nos asegura que cada convergente con denominador Q es la mejor aproximación racional a α con denominadores menores o iguales que Q.

Teorema 4.25 *Supongamos que $n \geq 1$. Si $\frac{P_n}{Q_n} \neq \frac{P}{Q}$ y $1 \leq Q \leq Q_n$ entonces*

$$|Q_n\alpha - P_n| < |Q\alpha - P| \quad y \quad \left|\alpha - \frac{P_n}{Q_n}\right| < \left|\alpha - \frac{P}{Q}\right|.$$

Demostración. Sean

$$a = (-1)^n(QP_{n-1} - PQ_{n-1}) \quad \text{y} \quad b = (-1)^n(QP_n - PQ_n).$$

Por hipótesis a y b son enteros no nulos. Entonces

$$\begin{aligned} aP_n - bP_{n-1} &= (-1)^n(QP_{n-1}P_n - PQ_{n-1}P_n - QP_nP_{n-1} + PQ_nP_{n-1}) \\ &= P(-1)^n(Q_nP_{n-1} - Q_{n-1}P_n) = P. \end{aligned}$$

y análogamente

$$aQ_n - bQ_{n-1} = Q.$$

Por otro lado tenemos

$$|Q_n\alpha - P_n| < \frac{1}{Q_{n+1}} < \frac{1}{Q_n}.$$

Si $Q = Q_n$ entonces $P \neq P_n$ y

$$\begin{aligned} |Q\alpha - P| &= |Q_n\alpha - P_n + P_n - P| \geq |P_n - P| - |Q_n\alpha - P_n| \\ &> |P_n - P| - \frac{1}{Q_n} = \frac{|P_n - P|Q_n - 1}{Q_n} > \frac{1}{Q_n} > |Q_n\alpha - P_n|. \end{aligned}$$

Supongamos que $Q \neq Q_n$. Como $1 \leq aQ_n - bQ_{n-1} = Q < Q_n$ y a y b son enteros no nulos, a y b tienen el mismo signo. Además, de (4.3) deducimos que $Q_n\alpha - P_n$ y $Q_{n-1}\alpha - P_{n-1}$ tienen signos opuestos, lo que implica que $a(Q_n\alpha - P_n)$ y $-b(Q_{n-1}\alpha - P_{n-1})$ tienen el mismo signo y el último no es cero porque existe en n-ésimo convergente y por tanto $\alpha \neq \frac{P_{n-1}}{P_n}$. Luego

$$|Q_n\alpha - P_n| < |a(Q_n\alpha - P_n) - b(Q_{n-1}\alpha - P_{n-1})| = |Q\alpha - P|.$$

Esto demuestra la primera desigualdad y en realidad también la segunda pues:

$$\left|\alpha - \frac{P_n}{Q_n}\right| = \frac{1}{Q_n}|Q_n\alpha - P_n| < \frac{1}{Q_n}|Q\alpha - P| = \frac{Q}{Q_n}\left|\alpha - \frac{P}{Q}\right| \leq \left|\alpha - \frac{P}{Q}\right|.$$

∎

De hecho los convergentes son los únicos elementos que satisfacen la condición del Teorema 4.26:

Teorema 4.26 (*Teorema de mejor aproximación*) [2] *Sea α un número real positivo y sea $\frac{p}{q}$ una fracción racional irreducible. Entonces las siguientes condiciones son equivalentes:*

(1) $\frac{p}{q}$ es uno de los convergentes de α.

(2) Para toda fracción irreducible $\frac{a}{b} \neq \frac{p}{q}$ con $0 \leq b \leq q$ se verifica $|q\alpha - p| < |b\alpha - a|$.

Corolario 4.27 (Teorema de Aproximación de Dirichlet) *Sea α un número real positivo y sea $\frac{p}{q}$ una fracción racional irreducible positiva. Si $\left|\alpha - \frac{p}{q}\right| < \frac{1}{2q^2}$ entonces $\frac{p}{q}$ es uno de los convergentes de α.*

Demostración. Supongamos que $\left|\alpha - \frac{p}{q}\right| < \frac{1}{2q^2}$. Si $\frac{p}{q}$ no es uno de los convergentes de α. Entonces, del Teorema de mejor aproximación (Teorema 4.26) deducimos que existe una fracción irreducible $\frac{a}{b} \neq \frac{p}{q}$ con $0 \leq b \leq q$ y $|q\alpha - p| \geq |b\alpha - a|$. Luego

$$\left|\alpha - \frac{a}{b}\right| \leq \frac{1}{b}|b\alpha - a| \leq \frac{1}{b}|q\alpha - p| \leq \frac{q}{b}\left|\alpha - \frac{p}{q}\right| \leq \frac{q}{b}\frac{1}{2q^2} = \frac{1}{2bq}.$$

Por tanto

$$\left|\frac{aq - bp}{bq}\right| = \left|\frac{a}{b} - \frac{p}{q}\right| = \left|\frac{a}{b} - \alpha + \alpha - \frac{p}{q}\right| \leq \left|\frac{a}{b} - \alpha\right| + \left|\alpha - \frac{p}{q}\right| \leq \frac{1}{2bq} + \frac{1}{2q^2}.$$

[2] http://www.math.binghamton.edu/dikran/478/Ch7.pdf. Capítulo 7 de los apuntes de Real Analysis, por D. Karagueuzian, Theorem 7.20.

Como $\frac{a}{b} \neq \frac{p}{q}$, se tiene que $aq - bp$ es un entero diferente de cero, con lo cual $1 \leq |aq - bp|$. Por tanto, $\frac{1}{bq} < \frac{1}{2bq} + \frac{1}{2q^2}$, con lo cual $\frac{1}{2bq} < \frac{1}{2q^2}$. Luego $q < b$ lo que nos da una contradicción. Concluimos que $\frac{p}{q}$ es uno de los convergentes de α. ∎

Vamos con la aplicación al criptoanálisis de RSA que es el Ataque de Weiner para exponente bajo. Supongamos que (n, d) es una clave privada para RSA que satisface satisfaga $d \leq \frac{\sqrt[4]{n}}{3}$ y $n = pq$, con p y q primos con $p < q < 2p$. El nombre de ataque para exponente bajo se basa en que la función de descifrado está dada por $x \mapsto x^d \mod n$. Sea (n, c) la clave pública. Entonces $cd = 1 + t\phi(n)$ para algún entero t. Además $1 \leq c, d \leq \phi(n)$, con lo que $\frac{t}{d}\phi(n) = \frac{cd-1}{d} \leq \phi(n) - \frac{1}{d} < \phi(n)$ y, por tanto, $0 \leq t \leq d \leq \frac{\sqrt[4]{n}}{3}$. Si $t = 0$ entonces $c = d = 1$ y suponemos que ese no es el caso porque en tal caso no habría que trabajar nada para descubrir la clave privada. Entonces

$$\begin{aligned}\left|\frac{c}{n} - \frac{t}{d}\right| &= \left|\frac{cd - tn}{nd}\right| = \left|\frac{1 + t(\phi(n) - n)}{nd}\right| = \left|\frac{1 + t(1 - p - q)}{nd}\right| \\ &= \frac{1 + t(p + q - 1)}{nd} \leq \frac{t(p + q)}{nd} \leq \frac{3tp}{nd} < \frac{3t\sqrt{n}}{nd} \leq \frac{\sqrt[4]{n}}{d\sqrt{n}} = \frac{1}{d\sqrt[4]{n}} < \frac{1}{3d^2}.\end{aligned}$$

Del Teorema de Aproximación de Dirichlet (Corolario 4.27) deducimos que $\frac{t}{d}$ es uno de los convergentes de $\frac{c}{n}$. Por tanto podemos intentar buscar la fracción entre los convergentes de $\frac{c}{n}$. Vamos a ver cómo hacer esto. Pongamos $C = \frac{t}{d}$, que será uno de los convergentes. Entonces

$$\phi(n) = \frac{cd - 1}{t} = \frac{cd}{t} - \frac{1}{t} = \frac{c}{C} - \frac{1}{t}.$$

Como $\phi(n)$ es un entero positivo y suponemos que $t > 1$, se tiene que

$$n - p - q + 1 = \phi(n) = \left\lfloor \frac{c}{C} \right\rfloor.$$

Por tanto, si $h = n + 1 - \left\lfloor \frac{c}{C} \right\rfloor$ entonces p y q son las soluciones del sistema de ecuaciones

$$n = pq, \quad p + q = h.$$

O lo que es lo mismo las raíces del polinomio $X^2 - hX + n$. Eso implica que $h^2 - 4n = m^2$ para algún entero m y tendremos que $n = \frac{h^2 - m^2}{4} = \frac{h-m}{2} \cdot \frac{h+m}{2}$, o sea

$$p = \frac{h - m}{2} \quad \text{y} \quad q = \frac{h + m}{2}. \tag{4.5}$$

Por tanto, lo que haremos será probar con C entre los convergentes de $\frac{c}{n}$ hasta $(\lfloor n + 1 - \frac{c}{C} \rfloor)^2 - 4n$ sea un cuadrado perfecto. Una vez conseguido calcularemos su raíz cuadrada m y con este valor calcularemos p y q.

Ejemplo 4.28 Consideremos la clave pública $(n = 248696658407, c = 90538917193)$ de RSA. Comenzamos calculando la fracción continua de $\frac{n}{d}$

```
gap> n:=248696658407;
248696658407
gap> c:=90538917193;
90538917193
gap> x:=Indeterminate(Integers);
x_1
gap> ContinuedFractionExpansionOfRoot(n*x-c,100);
[ 0, 2, 1, 2, 1, 19, 12, 1, 1, 4, 1, 25, 8, 2, 29, 8, 2, 4, 1, 6 ]
gap> Length(last);
20
```

Con ella podemos calcular la lista de los convergentes, aunque en realidad GAP nos la da directamente.

```
gap> conv := List([1..20],i->ContinuedFractionApproximationOfRoot(n*x-c,i));
[ 0, 1/2, 1/3, 3/8, 4/11, 79/217, 952/2615, 1031/2832, 1983/5447, 8963/24620,
  10946/30067, 282613/776295, 2271850/6240427, 4826313/13257149,
  142234927/390697748, 1142705729/3138839133, 2427646385/6668376014,
  10853291269/29812343189, 13280937654/36480719203, 90538917193/248696658407 ]
```

Por supuesto el primer convergente 0 no nos resulta útil. Tomando el segundo convergente $c = \frac{1}{2}$, calculamos $(\lfloor n + 1 - \frac{c}{C} \rfloor)^2 - 4n$ y miramos a ver si es un cuadrado perfecto.

```
gap> C:=conv[2];
1/2
gap> h:=n+1-Int(c/C);
67618824022
gap> m := RootInt(h^2-4*n);
67618824014
gap> m^2=h;
false
```

Como no lo es, seguimos probando con los siguientes convergentes hasta que $(\lfloor n + 1 - \frac{c}{C} \rfloor)^2 - 4n$ sea un cuadrado perfecto.

```
gap> i:=3;;C:=conv[i];h:=n+1-Int(c/C);;m:=RootInt(h^2-4*n);;m^2=h^2-4*n;
1/3
false
gap> i:=i+1;;C:=conv[i];h:=n+1-Int(c/C);;m:=RootInt(h^2-4*n);;m^2=h^2-4*n;
3/8
false
gap> i:=i+1;;C:=conv[i];h:=n+1-Int(c/C);;m:=RootInt(h^2-4*n);;m^2=h^2-4*n;
4/11
false
gap> i:=i+1;;C:=conv[i];h:=n+1-Int(c/C);;m:=RootInt(h^2-4*n);;m^2=h^2-4*n;
79/217
true
```

Lo hemos obtenido en el sexto convergente que es $\frac{79}{217}$. Calculando

```
gap> p:=(h-m)/2;
287921
gap> q:=(h+m)/2;
863767
gap> p*q=n;
true
```

Obtenemos la factorización

$$n = 287921 \cdot 863767.$$

Además estamos seguros que la clave privada es $(n, d = 217)$ y que $cd - 1 = \phi(n)$:

```
gap> d:=DenominatorRat(C);
217
gap> t:=NumeratorRat(C);
79
gap> phi:=(p-1)*(q-1);
248695506720
gap> c*d-1=t*phi;
true
```

4.7 Criba cuadrática

Una de las partes más lentas del método de factorización con bases de factores es intentar factorizar los elementos de la forma $(b^2 \mod n)$. En esta sección vamos a introducir otra variación del método de factorización con bases de factores llamado *Criba Cuadrática* que por un lado selecciona los enteros b para que $(b^2 \mod n)$ sea igual a $b^2 - n$, por otro lado selecciona la base de factores que puedan aparecer en la factorización de números de esta forma y por último utiliza una estrategia para evitar divisiones que no vayan a resultar efectivas.

En el resto de la sección n es un entero positivo con lo que $(b^2 \mod n) \in \left[\frac{1-n}{2}, \frac{n-1}{2}\right]$.

Elección de los enteros b

Como hemos dicho arriba la primera parte de la estrategia es utilizar números b con $(b^2 \mod n) = b^2 - n$. Para conseguir esto sea $r = \lfloor \sqrt{n} \rfloor$, o sea r es el único entero con $r^2 \leq n < (r+1)^2$. Sea i un entero tal que

$$\frac{(1-\sqrt{2})r + 1}{\sqrt{2}} \leq i \leq \left(\sqrt{\frac{3}{2}} - 1\right) r. \tag{4.6}$$

y tomemos $b_i = r + i$. Entonces

$$\frac{r+1}{\sqrt{2}} \leq b_i = r + i \leq \sqrt{\frac{3}{2}} r$$

entonces

$$\frac{n}{2} < \frac{(r+1)^2}{2} \leq b_i^2 \leq \frac{3r^2}{2} \leq \frac{3n}{2}$$

y por tanto

$$(b_i^2 \mod n) = b_i^2 - n.$$

Por tanto vamos a considerar los enteros de la forma $b_i = r + i$ con i satisfaciendo (4.6) y para cada tal i ponemos $c_i = b_i^2 - n$. Obsérvese que c_i tiene el mismo signo que i y su valor absoluto crece con el de i con lo, para aumentar las posibilidades de poder factorizar c_i vamos tomando los i en valor creciente en valor absoluto es decir seguiremos la secuencia $i = 0, 1, -1, 2, -2, \ldots$.

Elección de la base de factores

Deseamos factorizar los números de la forma $c_i = b_i^2 - n$. Si p es un primo que divide a c_i entonces $n \equiv b_i^2 \mod p$. Por tanto n es un resto cuadrático módulo p. Por tanto solo resultarán útiles los primos que cumplan esta condición. Por tanto vamos a elegir nuestra base de factores $B = \{p_1, \ldots, p_k\}$ formada solo por los primeros primos, hasta una cantidad razonable, para los cuales n es un resto cuadrático. Recordemos que podemos utilizar el Algoritmo 9 para decidir si un número es resto cuadrático módulo cada p_i.

Por ejemplo, supongamos que queremos factorizar el número $n = 3234734999716657241639$. Construimos una base de factores formada los primos $p \leq 2000$ para los cuales n es resto cuadrático módulo p que resultan ser los siguientes:

2	5	7	13	19	31	41	43	47	53	61	67	79	89
101	103	107	113	137	139	149	151	163	167	173	193	211	223
233	251	269	281	293	311	317	337	349	367	389	397	421	431
443	449	457	461	463	499	509	541	563	569	577	593	599	601
607	617	659	709	719	727	733	739	751	761	769	773	787	811
823	827	857	859	863	877	887	911	919	941	947	953	971	977
983	991	997	1009	1013	1021	1033	1049	1061	1063	1087	1093	1109	1129
1151	1153	1171	1213	1217	1223	1259	1277	1279	1283	1291	1297	1307	1321
1327	1427	1429	1439	1447	1459	1471	1481	1483	1499	1511	1523	1531	1543
1549	1553	1559	1571	1609	1613	1621	1627	1699	1721	1723	1741	1753	1787
1801	1823	1871	1879	1901	1907	1931	1949	1973	1979	1987	1993	1997	1999

Cálculo de restos cuadráticos de n módulo los primos de la base de factores

El siguiente paso consiste en construir una lista

$$R = (r_j : j = 1, \ldots, k)$$

de forma que para cada j se cumpla:

$$r_j^2 \equiv n \mod p_j.$$

Para ello podemos utilizar el Algoritmo 14 que encontrará las soluciones en tiempo polinomial probabilístico (de hecho polinomial si se dispone de no restos cuadráticos). En realidad salvo que $p \equiv 3 \mod 8$ se puede encontrar una raíz cuadrada de n módulo p de forma directa aplicando el método del Problema 15.

En nuestro ejemplo la lista R es la siguiente:

1	2	4	9	4	20	33	11	37	13	43	24	67	47
56	9	56	67	133	91	73	84	36	114	84	105	176	16
80	198	118	173	206	32	12	56	313	258	125	8	27	128
208	171	402	179	169	307	303	243	28	114	23	22	421	78
176	345	350	489	668	218	402	581	376	28	458	571	353	391
482	367	786	833	70	636	808	571	679	358	149	588	655	277
610	132	679	340	904	737	272	307	1002	386	591	622	350	560
199	926	246	581	62	588	879	1245	916	1061	430	132	929	618
79	233	644	256	688	267	95	411	926	209	95	991	729	124
516	1350	492	1063	911	389	905	280	947	243	776	879	1709	924
659	489	136	962	502	1811	1410	862	692	423	1630	1648	1113	1695

Factorización de s_i.

Ahora vamos a utilizar la lista R_i para factorizar cada uno de los c_i y lo que queremos conseguir es dar un método para saber para qué primos p_j de la base de factores dividen a c_i para solo intentar dividir por ellos. Para ello construimos dos copias de la misma lista que denotamos de forma diferente porque más adelante serán dos listas diferentes:

$$S_1 = ((x_{1,j}, y_{1,j}) : j = 1, \ldots, k) \quad S_{-1} = ((x_{-1,j}, y_{-1,j}) : j = 1, \ldots, k)$$

donde

$$x_{1,j} = x_{-1,j} = (r - r_j \mod p_j) \quad \text{e} \quad y_{1,j} = y_{-1,j} = (r + r_j \mod p).$$

Veamos qué haríamos con el primer c_i, o sea con $c_0 = b_0^2 - n = r^2 - n$. Se tiene que p_j divide a c_0 si y solo si $r_j^2 \equiv n \equiv r^2 \mod p_j$ si y solo si

$$r_j \equiv r \mod p_j \quad \text{ó} \quad r_j \equiv -r \mod p_j$$

si y solo si

$$x_i y_i = 0.$$

Por tanto solo tenemos que intentar dividir s_i por los primos p_j para los que $x_j y_j = 0$. O sea guardamos en una lista los p_j que cumplen esta condición e vamos dividiendo s_i por cada uno de estos primos. Si conseguimos factorizar s_i de esta manera guardaremos b_i, c_i y el vector binario $\varepsilon_i = \varepsilon(b_i)$ como se explicó en la Sección 4.5.

El resto del algoritmo sigue más o menos de la misma manera con los diferentes b_i y c_i. Sólo tenemos que preocuparnos de actualizar las lista S_1 y S_{-1} en cada paso. Recuérdese que los b_i van tomando los valores $r + i$ con $i = 0, 1, -1, 2, -2, \ldots$. Una vez que hemos completado los pasos i y $-i$ (que en el caso $i = 0$ es solo un elemento) actualizamos las listas S_1 y S_{-1} poniendo

$$\begin{array}{rclcrcl} x_{1,j} & := & (x_j + 1 \mod p_j), & & y_{1,j} & := & (y_j + 1 \mod p_j) \\ x_{-1,j} & := & (x_{-1,j} - 1 \mod p_j), & & y_{-1,j} & := & (y_{-1,j} - 1 \mod p_j). \end{array}$$

Obsérvese que en cada paso

$$x_{1,j} \equiv b_i - r_j \mod p_j, \quad y_{1,j} \equiv b_i + r_j \mod p_j$$

y

$$x_{-1,j} \equiv b_{-i} - r_j \mod p_j, \quad y_{1,j} \equiv b_{-i} + r_j \mod p_j$$

Por tanto razonando como en el párrafo anterior observamos que p_j divide a s_i si y solo $x_{1,j}y_{1,j} = 0$. Similarmente p_j divide a s_{-i} si y solo si $x_{-1,j}y_{-1,j} = 0$.

El algoritmo seguiría el buscando ternas $(b_i, c_i = b_i^2 - n, \epsilon_i)$ para las que c_i se pueda factorizar con la base de factores y, como se explicó en la Sección 4.5 cuando se tenga una relación de dependencia lineal entre los vectores binarios ϵ_i se podrán obtener dos enteros $b = \prod_{i\in X} b_i$ y $c = \sqrt{\prod_{i\in X} c_i}$ de forma que $b^2 \equiv c^2 \mod n$, donde X está formado por los i que aparezcan en la relación de dependencia lineal entre los ϵ_i.

Se buscan relaciones de dependencia de este tipo hasta que o $\mcd(b+c, n)$ ó $\mcd(b-c, n)$ sea un divisor propio de n.

Para nuestro número n el algoritmo proporcionó $b = 1119533385483820437649$, y $c = 1555456745345335150292$ que satisfacen $\mcd(b + c, n) = 34359738421$ y $\mcd(b - c, n) = 94143178859$, que son dos primos cuyo producto es n.

4.8 Tareas

(1) Hacer un programa en el que se utilice una lista de primos pequeños (por ejemplo menores o iguales que un millón) que se pueden haber obtenido con la Criba de Eratóstenes, y que calcule la factorización de un número como producto de los elementos de esa lista, si eso es posible.

(2) Programar el Método de Fermat para encontrar un divisor próximo a la raíz cuadrada.

(3) Programar los métodos de Pollard, $(p-1)!$ y ρ. En ambos casos es conveniente poner entre los parámetros de entrada la semilla inicial y un tope para el número de ciclos.

(4) Hacer un programa que dado un número real positivo α y un número natural n calcule la fracción continua de longitud n asociada a α y el convergente P_n/Q_n.

(5) Programar el Algoritmo 20.

(6) Programar la Criba Cuadrática.

(7) Aplicar los programas anteriores a números cada vez más grandes. ¿Hasta qué tamaño los algoritmos funcionan siempre bien?

(8) Sea $M \leq 2\sqrt{n}$. Demostrar que la media de los números de la forma $|r^2 - n|$, para $|r - \sqrt{n}| < M$ es $\sqrt{n}M$.

(9) Sean p un primo impar y a un entero coprimo con p. Demostrar que si la ecuación $x^2 \equiv a \mod p^n$ tiene solución, entonces la ecuación $x^2 \equiv a \mod p^{n+1}$ también tiene solución.

(10) Sea n un entero positivo que no es un cuadrado perfecto. Construimos de forma recursiva seis sucesiones $(a_i), (b_i), (c_i), (e_i), (p_i), (q_i)$ poniendo:

$$e_0 = \sqrt{n}, \quad a_0 = \lfloor e_0 \rfloor, \quad b_{-1} = 0, \quad b_0 = a_0, \quad c_{-1} = 1, \quad c_0 = n - a_0^2,$$
$$p_{-1} = 1, \quad p_0 = a_0, \quad q_0 = 0, \quad q_1 = 1;$$

y para $i \geq 1$:

$$e_i = \frac{a_0+b_{i-1}}{c_{i-1}}, \quad a_i = \lfloor e_i \rfloor, \quad b_i = a_i c_{i-1} - b_{i-1}, \quad c_i = c_{i-2} + a_i(b_{i-1} - b_i)$$
$$p_i = p_{i-2} + a_i p_{i-1}, \quad q_i = q_{i-2} + a_i q_{i-1}.$$

Demostrar que para cada $i \geq 0$ se verifica:

(a) $n = b_i^2 - c_{i-1}c_i$.
(b) $a_i + e_i = \frac{1}{e_{i+1}}$.
(c) $\sqrt{n} = [a_0, a_1, \ldots, a_{n-1}, e_n] = [a_0, a_1, a_n, \ldots]$.
(d) Si $i \geq 3$, entonces $e_i = \frac{c_{i-3}}{\sqrt{n}-b_{i-2}}$.
(e) $0 < a_i < 2\sqrt{n}$, $0 < b_i < \sqrt{n}$, $0 < c_i < 2\sqrt{n}$.
(f) La sucesiones $(a_i), (b_i), (c_i)$ y (e_i) son periódicas.
(g) $\sqrt{n} = \frac{P_{i-1}+e_i P_i}{Q_{i-1}+e_i Q_i}$.
(h) $P_i^2 - nQ_i^2 = (-1)^i C_i$.
(i) Existe un entero m para el que la ecuación $x^2 - ny^2 = m$ tiene infinitas soluciones enteras.
(j) La ecuación diofántica $x^2 - ny^2 = 1$ tiene solución. (Indicación: Utilizar el apartado anterior para mostrar que hay dos parejas de enteros (x_1, y_1) y (x_2, y_2) que son soluciones de $x^2 - ny^2 = m$, con $x_1 \equiv x_2 \equiv m$ e $y_1 \equiv y_2 \mod m$. Entonces los enteros u y v definidos por $u + v\sqrt{n} = (x_1 + y_1\sqrt{n})(x_2 - y_2\sqrt{n})$ son múltiplos de m y satisfacen $u^2 - nv^2 = m^2$.)

(11) Demostrar que si α es un número real, entonces la fracción continua infinita de α es periódica si y sólo si α es una raíz de un polinomio de grado 2 con coeficientes enteros.

(12) Mostrar que los numeradores y denominadores de los convergentes de la fracción continua $[1, 1, 1, \ldots]$ son los elementos de la sucesión de Fibonacci. Calcular $\alpha = [1, 1, 1, \ldots]$.

(13) Utilizar el ataque de Weiner para exponente bajo para calcular la clave privada de la siguiente clave pública (n, c) de RSA con $n = 98262346432813563076125474695015891 51$ y $c = 71377371189015895030755398514593512 99$. Implementar el ataque de Weiner en un programa informático.

(14) Demostrar que el Algoritmo 14 es de tiempo polinomial. Implementarlo.

(15) Sea p un primo tal que $p \equiv 1 \mod 4$ y sea n es un resto cuadrático módulo p, coprimo con p. Demostrar las siguientes afirmaciones:

(a) Si $b = (n^{\frac{p-1}{4}} \mod p)$ entonces $b = \pm 1$.
(b) Si $p \equiv 5 \mod 8$ y

$$a = \begin{cases} n^{\frac{p+3}{8}} \mod p, & \text{si } b = 1; \\ (4n)^{\frac{p+3}{8}} \frac{p+1}{2} \mod p, & \text{si } b = -1; \end{cases}$$

entonces a es una raíz cuadrada de n módulo p.

Capítulo 5

Logaritmo discreto

La principal desventaja del criptosistema RSA consiste en la necesidad de encontrar primos grandes que resistan los métodos de factorización del capítulo anterior. Por contra los criptosistemas cuya fortaleza se basa en el Problema del Logaritmo Discreto necesitan semigrupos como plataforma para las operaciones criptográficas. En este capítulo vamos a estudiar algunos problemas para tratar el problema del logaritmo discreto. Por simplicidad nos restringiremos al caso en que el problema se plantea en un grupo. Vamos a comenzar estableciendo alguna notación básica.

Dado un elemento g de orden n de un grupo la aplicación $\exp_g : \mathbb{Z}_n \to \langle g \rangle$, dada por $\exp_g(x) = g^x$, es un isomorfismo cuyo inverso denotamos por $\log_g$ y llamaremos *logaritmo* en *base* g. Obsérvese que si $x = \log_g(a)$ e $y = \log_g(b)$ entonces $g^{x+y} = g^x g^y = ab$, con lo que $\log_g(ab) = \log_g(a) + \log_g(b) \mod n$. El Problema del Logaritmo Discreto es el del cálculo de $\log_g(h)$ para un $h \in \langle g \rangle$. En diversos criptosistemas se fija un elemento g de orden finito de forma que la información en claro es un número y la información cifrada es una potencia de g. Por tanto, el problema de romper dicho criptosistema consiste en resolver ecuaciones del tipo $g^X = h$, o lo que es lo mismo en calcular $\log_g(h)$, para g fijado y diversos valores de h. Fijaremos pues un elemento g de un grupo y en algunos algoritmos que plantearemos habrá una fase inicial que llamaremos "precálculo" que sólo depende de la base g. La salida del precálculo será útil para resolver muchas ecuaciones $g^X = h$, con g fijo y h variable.

En principio se puede plantear el Problema del Logaritmo Discreto en cualquier grupo, sin embargo los grupos en los que trabajemos deben satisfacer las siguientes condiciones:

(1) Ha de ser fácil codificar los elementos del grupo de forma que se puedan realizar operaciones con las representaciones codificadas de estos elementos.

(2) El Problema del Logaritmo Discreto en el grupo no ha de tener una solución fácil. En particular, es necesario utilizar grupos con elementos de orden grande.

La última condición sugiere utilizar grupos cíclicos de orden grande. Una primera idea naïf podría ser utilizar el grupo aditivo de $\mathbb{Z}_n$. Sin embargo, resolver el Problema del Logaritmo Discreto en $\mathbb{Z}_n$ equivale a resolver una ecuación de congruencias $aX \equiv b \mod n$ lo cual es bastante fácil.

También podemos construir grupos cíclicos cogiendo grupos de unidades de cuerpos finitos. Estos son grupos que cumplen las dos condiciones anteriores siempre que el orden del cuerpo sea grande. Obsérvese que no necesitamos construir primos grandes para tener cuerpos finitos de orden grande pues para cada primo p y cada número natural n existe un cuerpo de orden $q = p^n$. De hecho, sólo existe uno, salvo isomorfismos, que hemos denotado como $\mathbb{F}_q$. Para construir $\mathbb{F}_q$, elegimos un polinomio irreducible f de $\mathbb{F}_p[X]$ que tenga grado n. Entonces $\mathbb{F}_q = \mathbb{F}_p[X]/(f)$ es un cuerpo de orden n. Cada elemento a de $\mathbb{F}_q$ está representado por un polinomio $A \in \mathbb{F}_p[X]$ de grado menor que n. Así identificamos los elementos de $\mathbb{F}_q^*$, con polinomios no nulos de grado menor que n (o con elementos de $\mathbb{F}_p^n \setminus \{0\}$) y multiplicamos como en $\mathbb{F}_p[X]$, y después reducimos módulo f. Esto nos da una codificación bastante simple de los elementos de $\mathbb{F}_q$. El problema de encontrar un polinomio irreducible en $\mathbb{F}_p[X]$ de grado predeterminado no es problema trivial pero no vamos a entrar en él.

Otros grupos bastante buenos como plataforma de criptosistemas basados en el Problema del Logaritmo Discreto son los asociados a curvas elípticas que estudiaremos en el Capítulo 6.

5.1 Pasos de Niño, Pasos de Gigante

Uno de los algoritmos más sencillos de cálculo del logaritmo discreto es el conocido con el nombre de Pasos de Niño, Pasos de Gigante. Para poder aplicar este algoritmo necesitamos conocer una cota superior del orden de la base del logaritmo.

Algoritmo 21. Pasos de Niño, Pasos de Gigante

ENTRADA: g, a elementos de un grupo y n un entero positivo (estimación de $|g|$).
$s := \lceil \sqrt{n} \rceil$.
$L :=$ Lista ordenada $\{ag^j : j = 0, \ldots, s-1\}$.
$c := g^s$ (que se puede calcular multiplicando el último elemento de L por $a^{-1}g$),
$i := 1$; $d := c$;
Mientras que $i \leq s$
  Si d aparece en L en la posición j-ésima entonces
    SALIDA: $is - j + 1$.
  $d := dc$; $i := i + 1$;
Si el algoritmo no se para en el bucle anterior entonces:
  SALIDA: "a no está en el grupo generado por g ó n es menor que el orden de g".

La primera parte del algoritmo, en la que se calcula la lista $L = \{ba^i : i = 0, \ldots, s-1\}$ es lo que recibe el nombre de Paso de Niño, pues se aumenta el exponente i de uno en uno. Los Pasos de Gigante aparecen en la segunda parte del algoritmo en la que se van calculando potencias de a^s.

Dejaremos que el lector justifique como ejercicio que el Algoritmo de Paso de Niño, Paso Gigante, resuelve de forma determinista el Problema el Logaritmo Discreto tanto en su versión de problema de decisión como de problema de búsqueda, siempre que la entrada n sea mayor o igual que el orden de g.

Ejemplo 5.1 Consideremos el número primo $p = 1039$. Como $p - 1 = 2 \cdot 3 \cdot 173$ es la factorización de $p - 1$ y

$$3^{\frac{p-1}{2}} \equiv -1 \mod p, \quad 3^{\frac{p-1}{3}} \equiv 140 \mod p, \quad \text{y} \quad 3^{\frac{p-1}{137}} \equiv 729 \mod p,$$

la clase del $a = 3$ módulo p es un generador de $\mathbb{Z}_p^*$. Pongamos $b = 782$ y apliquemos el algoritmo para el cálculo de $\log_3(b)$.

Como $s := \lceil \sqrt{p-1} \rceil = 33$, los pasos de niño proporcionan los siguientes valores de ba^i para $i = 0, 1, \ldots, s$:

$$L = \left\{ \begin{array}{c} 782, 268, 804, 334, 1002, 928, 706, 40, 120, 360, 41, 123, 369, 68, 204, 612, 797, \\ 313, 939, 739, 139, 417, 212, 636, 869, 529, 548, 605, 776, 250, 750, 172, 516 \end{array} \right\}.$$

En el Paso de Gigante empezamos calculando a^{33}. Obsérvese que como el último elemento de L ha de ser $c = ba^{s-1}$, en lugar de elevar a a 33, podemos calcular $a^{33} = b^{-1}ca \equiv 374 \mod p$. A partir de ahora iremos calculando potencias de 374 hasta encontrar un elemento de L. Estas potencias son

$$374, 650, 1013, 666, 763, 676, 347, 942, 87, 329, 444, 855, 797$$

Obsérvese que el primer elemento encontrado de L se ha obtenido después de hacer 13 pasos de gigante, con lo que $797 = a^{13s}$. Como además 797 ocupa la posición número 17 de los resultados obtenidos en los pasos de niño tenemos $ba^{16} = 797 = a^{13s}$, con lo que $b = a^{13s-16}$, o sea $\log_a(b) = 13s - 16 = 413$.

Por desgracia, el tiempo de cálculo de este algoritmo no es muy bueno ya que al menos tiene que hacer $\lceil \sqrt{n} \rceil$ pasos de niño, con lo que el tiempo es al menos de $O(\sqrt{n})$. Una estimación heurística sugiere que el tiempo de cálculo de Algoritmo de Paso de Niño, Paso de Gigante, no es mayor que esta cantidad.

5.2 Logaritmo discreto con ρ de Pollard

Pollard introdujo un método para el cálculo del logaritmo discreto que utiliza ideas similares a su algoritmo de factorización ρ. La idea consiste en buscar enteros γ, γ', α y α' para los que

$$g^\gamma a^\alpha = g^{\gamma'} a^{\alpha'}.$$

En tal caso $g^{\gamma-\gamma'} \equiv a^{\alpha'-\alpha} = g^{\log_g(a)(\alpha-\alpha')}$. Por tanto

$$(\alpha - \alpha') \log_g(a) \gamma - \gamma' \mod n,$$

donde $n = |g|$. O sea $\log_g(a)$ es una solución de la ecuación de congruencias

$$(\alpha - \alpha') X \gamma - \gamma' \mod n, \tag{5.1}$$

Sea $d = \gcd(n, \alpha - \alpha') = 1$. La ecuación de congruencias (5.1) tiene una única solución única módulo $\frac{n}{d}$. En particular, si $d = 1$ su única solución módulo n habrá de ser $\log_g(a)$. En cualquier

caso tendremos d posibles candidatos módulo n para $\log_g(a)$, con los que si d no es muy grande, tal vez podamos encontrar el valor exacto de $\log_g(a)$ probando todos los candidatos. Si esto no fuera suficiente para descubrir $\log_g(a)$, podemos buscar otros cuatro enteros $\gamma_1, \gamma_1', \alpha_1, \alpha_1'$ que proporcionen otras restricciones con la esperanza que, cuando tengamos suficientes restricciones en términos ecuaciones de congruencias del tipo de la de (5.1), el conjunto de soluciones sea único módulo n.

La forma de buscar los enteros γ, γ', α y α' es, en cierto sentido similar, a la del proceso que se utiliza en la factorización ρ de Pollard. Lo que se va haciendo es generando dos sucesión de números enteros (γ_i) y (α_i) hasta que encontremos dos elementos distintos (γ_i, α_i) y (γ_j, α_j) de esta sucesión con $g^{\gamma_i} a^{\alpha_i} = g^{\gamma_j} a^{\alpha_j}$. Esta sucesión se va generando a partir de una partición de $G = \langle g \rangle = S_0 \cup S_1 \cup S_2$ del grupo generado por g y dos enteros γ_0, α_0 que servirán para inicializar el algoritmo. Con la partición se definen tres aplicaciones

$$f : G \to G, \quad \gamma : G \times \mathbb{Z}_n \to \mathbb{Z}_n, \quad \alpha : G \times \mathbb{Z}_n \to \mathbb{Z}_n$$

con las siguientes reglas:

$$f(x) = \begin{cases} gx, & \text{si } x \in S_0; \\ x^2, & \text{si } x \in S_1; \\ ax, & \text{si } x \in S_2. \end{cases} \quad \gamma(x,n) = \begin{cases} 1+n, & \text{si } x \in S_0; \\ 2n, & \text{si } x \in S_1; \\ n & \text{si } x \in S_2. \end{cases} \quad \alpha(x,n) = \begin{cases} n, & \text{si } x \in S_0; \\ 2n, & \text{si } x \in S_1; \\ n+1, & \text{si } x \in S_2. \end{cases}$$

Entonces se construyen las sucesiones (x_i), (γ_i) y (α_i) con la siguiente regla recursiva para $i \geq 1$:

$$x_0 = g^{\gamma_0} a^{\alpha_0}, \quad \gamma_i = h(x_{i-1}, \gamma_{i-1}), \quad \alpha_i = l(x_{i-1}, \alpha_{i-1}), \quad x_i = f(x_{i-1}).$$

Obsérvese que

$$x_i = g^{\gamma_i} h^{\alpha_i}.$$

Esto está claro para $i = 0$ y razonando por inducción tenemos que si $i \geq 1$ entonces

$$x_i = f(x_{i-1}) = \left.\begin{cases} gx_{i-1} = g^{\gamma_{i-1}+1} a^{\alpha_{i-1}} & (\text{si } x_{i-1} \in S_0) \\ x_{i-1}^2 = g^{2\gamma_{i-1}} a^{2\alpha_{i-1}} & (\text{si } x_{i-1} \in S_1) \\ ax_{i-1} = g^{\gamma_{i-1}} a^{\alpha_{i-1}+1} & (\text{si } x_{i-1} \in S_2) \end{cases}\right\} = g^{\gamma(x_{i-1},\gamma_{i-1})} a^{\alpha(x_{i-1},\alpha_{i-1})} = g^{\gamma_i} a^{\alpha_i}.$$

La partición $G = S_0 \cup S_1 \cup S_2$ hay que cogerla de forma que no esté relacionada con la estructura de grupo de G. Por ejemplo, si G es el grupo de unidades de $\mathbb{Z}_n$, podemos tomar $S_0 = \{i \in \mathbb{Z}_n^* : 0 \leq i < \frac{n}{3}\}$, $S_1 = \{i \in \mathbb{Z}_n^* : \frac{n}{3} \leq i < \frac{2n}{3}\}$ y $S_2 = \{i \in \mathbb{Z}_n^* : \frac{2n}{3} \leq i < n\}$.

La desventaja del Algoritmo de Pasos de Niño-Pasos de Gigante, es que requiere ir almacenando en memoria una gran cantidad de datos. Si tomamos el Método ρ de Pollard para el logaritmo discreto simplemente acumulando dos sucesiones (γ_i) y (α_i) hasta que encontremos $(\gamma_i, \alpha_i) \neq (\gamma_j, \alpha_j)$ con $g^{\gamma_i} a^{\alpha_i} = g^{\gamma_j} a^{\alpha_j}$ también tendremos que utilizar mucha memoria. Pero aquí es donde se puede utilizar la misma idea que en la segunda versión de la factorización ρ de Pollard, es decir sólo comparar las pareja (γ_i, α_i) con la $(\gamma_{2^j}, \alpha_{2^j})$ con $2^j < i \leq 2^{j+1}$. De esta forma no tenemos que guardar en memoria la última pareja $(\gamma_{2^j}, \alpha_{2^j})$ y mientras que $2^j < i \leq 2^{j+1}$, comparar (γ_i, α_i) con $(\gamma_{2^j}, \alpha_{2^j})$, es decir ver si $g^{\gamma_i} a^{\alpha_i} = g^{\gamma_{2^j}} a^{\alpha_{2^j}}$.

5.3 Algoritmo de Silver, Pohlig y Hellman

Vamos a ver ahora una algoritmo para el cálculo de $\log_g$ suponiendo que no sólo conocemos el orden n de g sino también su factorización:

$$n = \prod_{i=1}^{t} p_i^{\alpha_i}.$$

El precálculo consiste en calcular para cada $i = 1, \ldots, t$ un elemento de de orden p_i en $\langle g \rangle$ los elementos de orden p_i de G que resultan ser

$$r_{i,j} = g^{j\frac{n}{p_i}}, \quad j = 1, \ldots, p_i - 1,$$

pues estamos suponiendo que g tiene orden n. Además ponemos

$$r_{i,0} = 1.$$

Ejemplo 5.2 Supongamos que G es el grupo multiplicativo del cuerpo con $q = 256 = 2^8$ elementos y g es un generador de G. Para empezar necesitaremos una forma de expresar los elementos de $\mathbb{F}_q$. Para ello construiremos $\mathbb{F}_q$ como $\mathbb{F}_q/(f)$ con f un polinomio de grado 8 irreducible sobre $\mathbb{F}_2$.

```
gap> x:=Indeterminate(GF(2));
x_1
gap> f:=x^8+x^4+x^3+x^2+One(GF(2));
x_1^8+x_1^4+x_1^3+x_1^2+Z(2)^0
gap> IsIrreducible(f);
true
gap> F:=AlgebraicExtension(GF(2),f);
<field of size 256>
gap> g:=RootOfDefiningPolynomial(F);
(a)
gap> Order(g);
255
```

Hemos elegido como polinomio irreducible $f = x^8 + x^4 + x^3 + x^2 + 1$ y hemos denotado por a la imagen de X por el homomorfismo natural $\mathbb{F}_2[X] \to \mathbb{F}_q = \mathbb{F}_2[X]/(f)$. Vamos a fijar a como la base, que resulta ser un elemento primitivo de $\mathbb{F}_q$, es decir un generador de $\mathbb{F}_q^*$.

Tenemos $q - 1 = 255 = 3 \cdot 5 \cdot 17$. Luego en la fase de precálculo tenemos que obtener los $r_{ij} = a^{j\frac{q-1}{p_i}}$ para $p_i = 3, 5, 17$ y $0 \leq j < p_i$.

```
gap> ps:=Factors(255);
[ 3, 5, 17 ]
gap> rs := [];
[  ]
```

```
gap> for i in [1..Length(ps)] do
>   p := ps[i];
>   gp := g^(n/p);
>   Add(rs,[gp^0]);
>   for j in [1..p-1] do
>     Add(rs[i],rs[i][j]*gp);
>   od;
> od;
gap> rs;
[ [ !Z(2)^0, (g+g^2+g^4+g^6+g^7), (Z(2)^0+g+g^2+g^4+g^6+g^7) ],
  [ !Z(2)^0, (g+g^3), (g^2+g^6), (g+g^4+g^7), (Z(2)^0+g^2+g^3+g^4+g^6+g^7) ],
  [ !Z(2)^0, (g+g^2+g^5), (g^5+g^6), (Z(2)^0+g^6+g^7),
      (Z(2)^0+g^3+g^4+g^5+g^7), (Z(2)^0+g+g^2+g^3),
      (Z(2)^0+g+g^2+g^3+g^4+g^6+g^7), (g+g^3+g^4), (Z(2)^0+g+g^3+g^4+g^5),
      (Z(2)^0+g^3+g^5+g^7), (Z(2)^0+g^2+g^4+g^6), (Z(2)^0+g^4+g^7),
      (g+g^2+g^4+g^7), (g^2+g^5+g^6), (Z(2)^0+g^3+g^4+g^6), (g^2+g^5),
      (g^2+g^3+g^5) ] ]
```

Obsérvese que los elementos de $\mathbb{F}_q$ están expresados como polinomios en a de grado menor que 8 y las listas de r_i son

$$\begin{aligned}
r_1 &= (1, g+g^2+g^4+g^6+g^7, 1+g+g^2+g^4+g^6+g^7)\\
r_2 &= (1, g+g^3, g^2+g^6, g+g^4+g^7, 1+g^2+g^3+g^4+g^6+g^7),\\
r_3 &= (1, g+g^2+g^5, g^5+g^6, 1+g^6+g^7, 1+g^3+g^4+g^5+g^7,\\
&\quad 1+g+g^2+g^3, 1+g+g^2+g^3+g^4+g^6+g^7, g+g^3+g^4, 1+g+g^3+g^4+g^5,\\
&\quad 1+g^3+g^5+g^7, 1+g^2+g^4+g^6, 1+g^4+g^7, g+g^2+g^4+g^7,\\
&\quad g^2+g^5+g^6, 1+g^3+g^4+g^6, g^2+g^5, g^2+g^3+g^5)
\end{aligned}$$

El objetivo de la fase de cálculo consiste en calcular $\log_g(a)$ para un elemento $a \in \langle g \rangle$. Por el Teorema Chino de los Restos, para conocer $\log_g(a)$, que es entero módulo n, basta con conocerlo módulo $p_i^{\alpha_i}$, para cada $i = 1, \ldots, l$. Fijamos $i = 1, \ldots, l$, ponemos $p = p_i$, $\alpha = \alpha_i$ y $r_j = r_{i,j}$ y vamos a determinar x módulo p^α. Esto equivale a encontrar enteros $0 \leq x_0, x_1, x_2, \ldots, x_{\alpha-1} < p$ tales que

$$x \equiv x_0 + x_1 p + x_2 p^2 + \ldots + x_{\alpha-1} p^{\alpha-1} \mod p^\alpha.$$

Veamos como podemos calcular uno detrás de otro $x_0, x_1, \ldots x_{\alpha-1}$. En primer lugar obsérvese que

$$a^{\frac{n}{p}} = g^{x_0 \frac{n}{p}} g^{n(x_1 + x_2 p + \cdots + x_{\alpha-1} p^{\alpha-2} + h p^{\alpha-1})} = r_{x_0}$$

Por tanto, para calcular x_0, simplemente elevamos a a $\frac{n}{p}$. El resultado tiene que ser igual a un único r_j con $0 \leq j < p$, que descubriremos comparando con la lista de $r_{i,j}$'s. Entonces $x_0 = j$. Una vez que conocemos x_0 podemos calcular ag^{-x_0} y elevarlo a $\frac{n}{p^2}$. El resultado será de nuevo uno de los r_{ij} pues

$$(ag^{-x_0})^{\frac{n}{p^2}} = g^{x_1 \frac{n}{p} + n(x_2 + x_3 p + \cdots + x_{\alpha-1} p^{\alpha-3} + h p^{\alpha-2})} = r_{x_1}.$$

Es decir, elevamos ag^{-x_0} a $\frac{n}{p^2}$ y lo comparamos con los r_{ij}, para decidir el valor de x_1. Supongamos que ya conocemos $x_0, \ldots, x_{k-1}$, con $k < \alpha$. Para obtener x_k, calculamos $ag^{-x_0-x_1p-\cdots-x_{k-1}p^{k-1}}$ elevado a $\frac{n}{p^{k+1}}$ y nuevamente obtenemos un $r_{i,j}$, concretamente r_{x_k} pues

$$(ag^{-x_0-x_1p-\cdots-x_{k-1}p^{k-1}})^{\frac{n}{p^{k+1}}} = g^{x_k\frac{n}{p}+(n)(x_{k+1}+x_{k+2}p+\cdots+x_{\alpha-1}p^{\alpha-k-2}+hp^{\alpha-k-1})} = r_{x_k}.$$

Ejemplo 5.3 Continuando con el Ejemplo 5.2, vamos a calcular $x = \log_g(a)$ con $a = 1 + g^2 + g^6 + g^7$. Empezamos calculando $x \mod 3$, para lo cual calculamos $a^{255/3}$

```
gap> a := One(GF(2))+g^2+g^6+g^7;
(Z(2)^0+g^2+g^6+g^7)
gap> a^(255/3);
!Z(2)^0
```

que resulta ser 1. Por tanto, $x_0 = 0$. Como la multiplicidad de 3 en 255 es 1, podemos pasar al siguiente primo y calcular $a^{255/5}$

```
gap> a^(255/5);
(g+g^4+g^7)
```

Como el resultado obtenido ocupa la cuarta posición en la lista r_2, tenemos $x \mod 5 = 3$. Con el último primo obtenemos que $a^{255/17}$ ocupa la quinta posición de r_3, con lo que $x \mod 17 = 4$.

```
gap> a^(255/17);
(Z(2)^0+g^3+g^4+g^5+g^7)
```

Resolviendo el sistema de congruencias

$$x \equiv 0 \mod 3, \quad x \equiv 3 \mod 5, \quad x \equiv 4 \mod 17$$

```
gap> ChineseRem([3,5,17],[0,3,4]);
123
```

deducimos que $\log_a(b) = 123$. Efectivamente

```
gap> g^123=a;
true
```

El método explicado se conoce con el nombre de Algoritmo de Silver, Pohlig y Helman.

Algoritmo 22. Algoritmo de Silver, Pohlig y Hellman

ENTRADA. g y h elementos de un grupo.

PRECÁLCULO

$n :=$ Orden de g;
$p_1^{\alpha_1} \cdots p_k^{\alpha_k} := n$ (Factorización de n en producto de primos);
Para cada $i = 1, \ldots, k$ y cada $j = 0, 1, \ldots, p_i - 1$: $r_{ij} := g^{j\frac{n}{p_i}}$.

CÁLCULO

Para cada $i = 1, \ldots, k$
  $p := p_i$; $\alpha := \alpha_i$; $(r_j)_{j=0}^{p-1} := (r_{ij})_{j=0}^{p-1}$; $z := y$.
  Para $l = 0, 1, \ldots, \alpha - 1$,
    $w := z^{\frac{n}{p^{l+1}}}$;
    Si $w \neq r_j$ para todo $0 \leq j < p$, entonces
      SALIDA: "h no está en el grupo generado por g"
    $x_l :=$ único $0 \leq j < p$ tal que $w = r_j$.
    $z := zg^{-x_l p^l}$.
$X_i := x_0 + x_1 p + \cdots + x_{\alpha-1} p^{\alpha-1}$.

SALIDA: Solución del sistema de congruencias

$$X \equiv X_1 \mod p_1^{\alpha_1}, \ldots, X \equiv X_k \mod p_1^{\alpha_k}.$$

Si consideramos el Algoritmo de Siver-Pohlig-Hellman en el caso en que el grupo plataforma es el grupo $\mathbb{F}_q^*$ de un cuerpo con $q = p^k$ elementos, con p primo, observamos que el orden de los elementos de $\mathbb{F}_q^*$ es un divisor de $q = p^k - 1$. Por tanto, nos interesa contar con métodos para factorizar números de la forma $p^n - 1$, con p primo. La siguiente proposición sirve para este propósito.

Proposición 5.4 *Sean n y m enteros positivos y sea p un divisor primo de $m^n - 1$. Entonces se verifica una de las dos siguientes condiciones:*

(1) $p | m^d - 1$, para algún divisor propio d de n.

(2) $p \equiv 1 \mod n$.

Si además p y n son impares, en el caso 2 se verifica $p \equiv 1 \mod 2n$.

Demostración. Como $m^n \equiv 1 \mod p$, se tiene $o_p(m)|n$. Por el Teorema de Fermat, $o_p(m)|p-1$. Por tanto, si $d = \text{mcd}(p-1, n)$, entonces $o_p(m)|d$, con lo que $m^d \equiv 1 \mod p$. Si $d < n$, se da el caso 1 y en caso contrario se da el caso 2. Si además p y n son impares entonces 2 divide a $p - 1$ y $\text{mcd}(2, n) = 1$, con lo que si $n|p - 1$, entonces $2n|p - 1$. ∎

Ejemplos 5.5 (1) Vamos a factorizar $2^{11} - 1 = 2047$. Sea p un divisor primo de 2047 que sea menor que $\sqrt{2047} < 46$. Por la Proposición 5.4, $p \equiv 1 \mod 22$, con lo que $p = 23$ ó $p = 45$. Calculando observamos que $2047 = 23 \cdot 89$.

(2) Vamos ahora a factorizar $2^{13} - 1 = 8191$. El mismo argumento muestra que los únicos factores primos posibles menores que la raíz cuadrada son 27, 53 y 79. Se comprueba que ninguno de estos es un divisor y por tanto 8191 es primo.

5.4 Cálculo de índices

En esta sección vamos a tratar el Problema del Logaritmo Discreto en los grupos de unidades de cuerpos finitos. En el caso en que el cuerpo sea $\mathbb{Z}_p$, con p primo, solemos identificar un elemento de $\mathbb{Z}_p$ con el entero que lo representa. Eso implica que la expresión $\log_g(a)$ con $g, a \in \mathbb{Z}_p$, puede dar lugar a error interpretando que se está calculando logaritmos habituales (en $\mathbb{R}^*$) en lugar de en $\mathbb{Z}_p$. Por tanto, se suele utilizar $\mathrm{Ind_g(a)}$ para denotar $\log_g(a)$ en $\mathbb{Z}_p^*$. Extendemos esta notación a cualquier cuerpo finito, con lo que si F es un cuerpo finito y $g, a \in F$ entonces $\mathrm{Ind_g(a)}$ denota un número n tal que $g^n = a$. Ese número se llama el *índice* a en la base g. Vamos a ver dos métodos para el cálculo de índices bajo la hipótesis de que la base g es un generador del grupo de unidades del cuerpo. Por fijar ideas $q = p^n$ con p primo y g es un generador de $\mathbb{F}_q$.

Cálculo de índices módulo un primo

En el primer método suponemos que $n = 1$, o sea $q = p$, con lo que vemos los elementos de $\mathbb{F}_p = \mathbb{Z}_p$ como enteros módulo p. Este método tiene ciertas similitudes con el método de factorización con bases de factores. Fijaremos una base de factores $B = \{p_1, \ldots, p_k\}$. En la fase de precálculo se buscan B-números de la forma $(g^s \mod p)$. Es decir, buscamos números $s_1, \ldots, s_l$ para los cuales se tenga

$$g^{s_i} \mod p = (-1)^{c_{i0}} p_1^{c_{i1}} \ldots p_k^{c_{ik}}.$$

Utilizando esto tendremos

$$\begin{array}{rcl} c_{10}\mathrm{Ind_g}(-1) + \mathrm{c_{11}Ind_g(p_1)} + \cdots + \mathrm{c_{1k}Ind_g(p_k)} & \equiv & s_1 \mod (p-1) \\ c_{20}\mathrm{Ind_g}(-1) + \mathrm{c_{21}Ind_g(p_1)} + \cdots + \mathrm{c_{2k}Ind_g(p_k)} & = & s_2 \mod (p-1) \\ & \cdots & \\ c_{l0}\mathrm{Ind_g}(-1) + \mathrm{c_{l1}Ind_g(p_1)} + \cdots + \mathrm{c_{lk}Ind_g(p_k)} & = & s_l \mod (p-1) \end{array}$$

Como $\mathrm{Ind_g}(-1) = \frac{\mathrm{p-1}}{2}$, si el sistema anterior es determinado módulo $p-1$, podremos calcular $\mathrm{Ind_g(p_1)}, \ldots, \mathrm{Ind_g(p_k)}$ resolviendo el sistema lineal de ecuaciones:

$$\begin{array}{rcl} c_{11}X_1 + \cdots + c_{1k}X_k & = & s_1 - c_{10}\frac{p-1}{2} \mod (p-1) \\ c_{21}X_1 + \cdots + c_{2k}X_k & = & s_2 - c_{20}\frac{p-1}{2} \mod (p-1) \\ & \cdots & \\ c_{l1}X_1 + \cdots + c_{lk}X_k & = & s_l - c_{k0}\frac{p-1}{2} \mod (p-1) \end{array}$$

Una vez que hayamos calculado $\mathrm{Ind_g(p_1)}, \ldots, \mathrm{Ind_g(p_k)}$ podemos plantearnos el cálculo de $\mathrm{Ind_g(h)}$ de la siguiente forma: Buscamos un entero s para el que $(hg^s \mod p)$ sea un B-número. Es decir, vamos calculando $(hg^s \mod p)$ para distintos enteros de s e intentando factorizar el resultado obtenido. Pongamos

$$hg^s = (-1)^{d_0} p_1^{d_1} \ldots p_k^{d_k} = g^{d_0 \frac{p-1}{2} + d_1 \mathrm{Ind_g(p_1)} + \cdots + \mathrm{d_k Ind_g(p_k)}},$$

y, por tanto,

$$\mathrm{Ind_g(h)} = (\mathrm{d_0}\frac{\mathrm{p-1}}{2} + \mathrm{d_1 Ind_g(p_1)} + \cdots + \mathrm{d_k Ind_g(p_k)} - \mathrm{s} \mod (\mathrm{p-1})).$$

Ejemplo 5.6 Mantengamos el primo $p = 1039$ y el elemento primitivo $a = 3$ de $\mathbb{F}_p$. Pongamos $b = 102$ y vamos a calcular $Ind_3(102)$. Observemos que $102 = 2{\cdot}3{\cdot}17$, y claramente $Ind_3(3) = 1$, con lo que si conseguimos calcular $Ind_3(2)$ y $Ind_3(17)$ habremos acabado. Calculando $(a^i \mod p)$ para valores pequeños de i observamos que $(a^{12} \mod p) = 2^9$, con lo que $9Ind_3(2) \equiv 12 \mod (p-1)$. Hay que tener cuidado con esto pues 9 no es coprimo con $p - 1 = 2 \cdot 3 \cdot 173$. Sin embargo, podremos dividir por 3 en la congruencia anterior y obtener que $3Ind_3(2) \equiv 4 \mod \frac{p-1}{3}$. Como el inverso de 3 módulo $\frac{p-1}{3}$ es 231, tenemos $Ind_3(2) = 4 \cdot 231 \equiv 232 \mod \frac{p-1}{3}$. Eso implica que $Ind_3(2)$ es congruente con 232, $232 + \frac{p-1}{3} = 578$ ó $232 + 2\frac{p-1}{3} = 924$. De hecho $3^{232} \equiv 2 \mod p$, o sea $Ind_3(2) = 232$. Ahora tenemos $(3^{84} \mod p) = 544 = 2^5 \cdot 17$. Por tanto, $84 = 5Ind_3(2) + Ind_3(17) \mod (p-1)$, o sea $Ind_3(17) = (84 - 5 \cdot 232 \mod (p-1)) = 1000$. Concluimos que $Ind_3(102) = (Ind_3(2) + Ind_3(3) + Ind_3(17) = 232 + 1 + 1000 \mod (p-1)) = 195$.

Reducción de cálculo de índices al de cálculo de índices módulo un primo

En el segundo método suponemos que se sabe calcular índices en $\mathbb{F}_p$. Nos referiremos a los elementos de $\mathbb{F}_p$ como constantes. El grupo $\mathbb{F}_p^*$, de constantes no nulas, es el único subgrupo de $\mathbb{F}_q^*$ de orden $p-1$ y por tanto $g' = g^{\frac{q-1}{p-1}}$ es un generador de $\mathbb{F}_p^*$. Obsérvese que la hipótesis de que se sabe calcular $\mathrm{Ind_g(a)}$ para a una constante es razonable si p es pequeño. En efecto, en tal caso, podemos calcular g'^i para todos los valores $i = 1, \ldots, p-1$. Como g' es generador de $\mathbb{F}_p^*$, los resultados proporcionan una tabla de parejas (i, g'^i), en el que la primera columna contiene todos los números de 1 a $p-1$ y la segunda todos los elementos de $\mathbb{F}_p^*$, o sea las constantes no nulas. Invirtiendo la posición de las columnas de esta tabla obtendremos una tabla en la que en la primera columna aparecen las constantes no nulas c y en la segunda columna aparece $\mathrm{Ind_{g'}(c)}$ para cada constante $c \neq 0$. Ahora aplicando la fórmula del cambio de base en logaritmos (ver Problema 1) obtenemos

$$\mathrm{Ind_g(c)} = \mathrm{Ind_g(g')Ind_{g'}(c)} = \frac{\mathrm{q-1}}{\mathrm{p-1}}\mathrm{Ind_{g'}(c)}.$$

Es decir, multiplicando por $\frac{q-1}{p-1}$ en la segunda columna de la tabla de logaritmos en base g' obtendremos la tabla de logaritmos en base g de las constantes no nulas.

Ejemplo 5.7 El polinomio $f = x^4 - x^2 - x + 2$ es irreducible sobre $\mathbb{F}_5$, por lo que lo utilizamos para construir $\mathbb{F}_{5^4}$. Además, si a es una raíz de f, entonces a tiene orden 624, con lo que a es un generador de $\mathbb{F}_{5^4}$.

```
gap> x:=Indeterminate(GF(5));
x_1
gap> f:=x^4-x^2-x+2*Z(5)^0;
x_1^4-x_1^2-x_1+Z(5)
gap> IsIrreducible(f);
true
gap> F:=AlgebraicExtension(GF(5),f);
<field of size 625>
gap> a:=RootOfDefiningPolynomial(F);
(a)
gap> Order(a);
624
```

Como $156 = \frac{5^4-1}{5-1}$, a^{156} es un generador de $\mathbb{F}_5$. En efecto $a^{156} = 2$, que es un generador de $\mathbb{F}_5$.

```
gap> a^156=2*Z(5)^0;
true
```

Concluimos que los logaritmos en base a de los elementos de $\mathbb{Z}_5$ son $\log_a(2) = 156$, $\log_a(4) = 312$, $\log_a(3) = 468$ y $\log_a(1) = 624$, que ponemos en la tabla

x	$\mathrm{Ind_a(x)}$
1	624
2	156
3	468
4	312

En este algoritmo el precálculo consiste en calcular los logaritmos de los elementos de un subconjunto B de $\mathbb{F}_q \setminus \{0, 1\}$ de cardinal h predeterminado. La elección de este cardinal es asunto de equilibrio. Cuanto más grande sea h más efectivo será el cálculo, sin embargo el precálculo tardará más en completarse. En la práctica se suele elegir B como el conjunto de los polinomios de grado menor o igual que m para un $m < n$ predeterminado y de forma que m no sea muy grande.

Ejemplo 5.8 Vamos a elegir $q = 2^7 = 128$ y construimos $\mathbb{F} = \mathbb{F}_{128}$ como $\mathbb{F}_2[X]/(x^7 + x + 1)$.

```
gap> x:=Indeterminate(F2);
x_1
```

```
gap> f:=x^7+x+Z(2);
x_1^7+x_1+Z(2)^0
gap> IsIrreducible(f);
true
gap> F:=AlgebraicExtension(GF(2),f);
<field of size 128>
gap> a:=RootOfDefiningPolynomial(F);
(a)
```

Los elementos de $\mathbb{F}$ son representados por polinomios en a de grado ≥ 6. Fijamos $B = \{a, 1+a\}$.

Una vez que se ha elegido el conjunto B de cardinal h, la parte del precálculo consiste en encontrar "suficientes" números t entre 1 y $q-1$, de forma que sepamos escribir a^t como producto de elementos de $\mathbb{F}_p$ y elementos de B.

Ejemplo 5.9 [Continuación del Ejemplo 5.8] Habíamos elegido $B = \{a, 1+a\}$. Vamos a calcular $a^i(1+a)^j$ para $-2 \leq i, j, \leq 2$. Denotaremos con H el conjunto de los elementos obtenidos de esta forma.

```
gap> B := [a,One(F)+a];
[ (a), (Z(2)^0+a) ]
gap> exp:=Cartesian([-2..2],[-2..2]);
[ [ -2, -2 ], [ -2, -1 ], [ -2, 0 ], [ -2, 1 ], [ -2, 2 ], [ -1, -2 ],
  [ -1, -1 ], [ -1, 0 ], [ -1, 1 ], [ -1, 2 ], [ 0, -2 ], [ 0, -1 ],
  [ 0, 0 ], [ 0, 1 ], [ 0, 2 ], [ 1, -2 ], [ 1, -1 ], [ 1, 0 ], [ 1, 1 ],
  [ 1, 2 ], [ 2, -2 ], [ 2, -1 ], [ 2, 0 ], [ 2, 1 ], [ 2, 2 ] ]
gap> H:=List(exp,x->B[1]^x[1]*B[2]^x[2]);
[ (Z(2)^0+a+a^3+a^6), (a+a^2+a^3+a^4+a^6), (Z(2)^0+a^5+a^6), (a^5),
  (a^5+a^6), (Z(2)^0+a^2+a^4), (Z(2)^0+a+a^2+a^3+a^4+a^5), (Z(2)^0+a^6),
  (a^6), (Z(2)^0+a+a^6), (a+a^3+a^5), (a+a^2+a^3+a^4+a^5+a^6), !Z(2)^0,
  (Z(2)^0+a), (Z(2)^0+a^2), (a^2+a^4+a^6), (Z(2)^0+a+a^2+a^3+a^4+a^5+a^6),
  (a), (a+a^2), (a+a^3), (Z(2)^0+a+a^3+a^5), (Z(2)^0+a^2+a^3+a^4+a^5+a^6),
  (a^2), (a^2+a^3), (a^2+a^4) ]
```

Ahora buscamos un entero $1 \leq t \leq 127$ tal que $a^t \in H$.

```
gap> t:=Random([1..127]);a^t in H;
35
false
gap> while not a^t in H do
> t:=Random([1..127]);
> od;
gap> t;
6
```

Obsérvese que hemos tenido que probar con varios valores de t hasta encontrar que $a^6 \in H$. La siguiente cuenta muestra que $a^6 = a^{-1}(1+a)$:

```
gap> Position(H,a^6);
9
gap> exp[9];
[ -1, 1 ]
gap> a^6=B[1]^-1*B[2];
true
```

Obsérvese que esto significa que $6 = \mathrm{Ind_a(a^6)} = -\mathrm{Ind_a(a)} + \mathrm{Ind_a(1+a)} = \mathrm{Ind_a(1+a)} - 1$, y por tanto $\mathrm{Ind_a(1+a)} = 7$.

```
gap> a^7;
(Z(2)^0+a)
```

El objetivo del precálculo es obtener $\mathrm{Ind_g(b)}$ para todos los elementos b de B. Pongamos $B = \{b_1, \ldots, b_h\}$. Entonces para cada entero t_i descubierto para el que g^{t_i} sea el producto de un elemento de $\mathbb{F}_p$ con un producto de elementos de B tendremos una relación $g^{t_i} = c_i b_1^{x_{i1}} + \cdots + b_h^{x_{ih}}$ con $c_i \in \mathbb{F}_p$ y $x_{i1}, \ldots, x_{ih}$ enteros que podemos considerar como elementos de $\mathbb{Z}_{q-1}$. Esto proporciona una relación de dependencia lineal

$$x_{i1}\mathrm{Ind_g(b_1)} + \mathrm{x_{i2}Ind_g(b_2)} + \cdots + \mathrm{x_{ih}Ind_g(b_h)} \equiv \mathrm{t_i} - \mathrm{Ind_g(c_i)} \mod (\mathrm{q}-1).$$

Es decir, $(\mathrm{Ind_g(b_1)}, \ldots, \mathrm{Ind_g(b_h)})$ es una solución de la siguiente ecuación lineal de congruencias

$$x_{i1}X_1 + x_{i2}X_2 + \cdots + x_{ih}X_h \equiv t_i - \mathrm{Ind_g(c_i)} \mod (\mathrm{q}-1).$$

Los "suficientes" valores se habrán obtenido cuando tengamos un sistema de ecuaciones lineales con solución única, es decir tales que la matriz de los x_{ij} tenga rango h módulo $q-1$.

Algoritmo 23. Cálculo Índices en característica pequeña. Precálculo

ENTRADA. g, un generador de $\mathbb{F}_q^*$.

[1] Se repite el siguiente proceso hasta completar una matriz de rango h módulo $q-1$:

$$A = (\alpha_{t,a})_{t\in I, a\in B} \in M_{m,h}(\mathbb{Z}_{q-1}).$$

Se elige al azar $1 \leq t \leq q-2$ y se determina si se puede escribir

$$g^t = c_t \prod_{a\in B} a^{\alpha_{c,a}} \text{ con } c_t \in \mathbb{F}_p \text{ y } \alpha_{t,a} \in \mathbb{Z}.$$

Si es así, se añade t al conjunto I y la fila $(\alpha_{t,a})_{a\in B}$ a la matriz.

[2] Se construye una tabla $\{(a, X_a) : a \in B\}$ con las soluciones del sistema lineal de congruencias:

$$t = \mathrm{Ind_g(c_t)} + \sum_{\mathrm{a\in B}} \alpha_{\mathrm{t,a}} \mathrm{X_a} \mod (\mathrm{q}-1).$$

Vamos a ver que el sistema que aparece en el precálculo tiene solución y que de hecho $X_a = \mathrm{Ind_g(a)}$, para todo $a \in B$. En efecto, supongamos que $Y_a = \mathrm{Ind_g(a)}$, es decir $g^{Y_a} = a$, para cada $a \in B$. Entonces para cada $t \in I$ se tiene

$$g^{\mathrm{Ind_g(c_t)}+\sum_{\mathrm{a\in B}} \alpha_{\mathrm{t,a}} \mathrm{Y_a}} = c_t \prod_{a\in B} a^{\alpha_{t,a}} = g^t.$$

Por tanto,

$$t \equiv \mathrm{Ind_g(c_t)} + \sum_{\mathrm{a\in B}} \alpha_{\mathrm{t,a}} \mathrm{Y_a} \mod (\mathrm{q}-1),$$

es decir, el sistema de congruencias tiene solución y la solución es única pues la matriz de los coeficientes tiene una submatriz invertible módulo $q-1$ de tamaño h.

Por tanto, en el precálculo habremos obtenido una tabla $\{(b, X_b) : b \in B\}$, de forma que $X_b = \mathrm{Ind_g(b)}$, para todo $b \in B$.

Para explicar la parte del cálculo volvemos al ejemplo

Ejemplo 5.10 [Continuación del Ejemplo 5.9] Una vez que tenemos $\mathrm{Ind_a(a)} = 1$ y $\mathrm{Ind_a(1+a)} = 7$ vamos a intentar calcular $\mathrm{Ind_a(1+a+a^3)}$. Para ello buscamos un número t tal que ya^t se pueda escribir como producto de elementos de B

```
gap> t:=Random([1..127]);
126
gap> while not y*a^t in H do
> t:=Random([1..127]);
> od;
```

```
gap> t;
80
gap> Position(H,y*a^t);
1
gap> exp[1];
[ -2, -2 ]
gap> B[1]^-2*B[2]^-2=y*a^t;
true
```

La cuenta anterior muestra que $ya^{80} = a^{-2}(1+a)^{-2}$. Por tanto

$$\mathrm{Ind_a(y)} = -80 - 2\mathrm{Ind_a(a)} - 2\mathrm{Ind_a(1+a)} = -96 \equiv 31 \mod 127$$

Efectivamente

```
gap> a^31;
(Z(2)^0+a+a^3)
```

En resumen, la parte de cálculo consiste en buscar un entero t entre 1 y $q-1$ para el que yg^t se pueda escribir como producto de un elemento de $\mathbb{F}_p$ y elementos de B y se utiliza una idea similar a la del precálculo para obtener $\mathrm{Ind_g(y)}$.

Algoritmo 24. Cálculo de Índices en característica pequeña. Cálculo

ENTRADA: $y \in \mathbb{F}_q^*$ y la tabla $\{(b, X_b) : b \in B\}$ obtenida en el precálculo.

CÁLCULO: Hasta conseguir obtener una respuesta positiva.
Elegimos $1 \leq t \leq q-2$ al azar;
$y_1 := yg^t$.
Intentamos encontrar $y_0 \in \mathbb{F}_p$ e $Y_a \in \mathbb{Z}$ tales que $y_1 = y_0 \prod_{a \in B} a^{Y_a}$ con .

SALIDA: $\mathrm{Ind_g(y_0) - t + \sum_{a \in B} Y_a X_a}$.

Utilizando este algoritmo se pueden calcular rápidamente logaritmos discretos en $\mathbb{F}_{2^n}$, para $n \leq 1000$.

5.5 Tareas

(1) Demostrar que la fórmula clásica del cambio de base en logaritmos también se verifica en el caso más general de logaritmos discretos, se decir

$$\log_g(a) = \log_g(g') \log_{g'}(a).$$

(2) Demostrar que el Algoritmo 21 resuelve de forma determinista el Problema del Logaritmo Discreto tanto en las versiones de problema de decisión y de problema de cálculo.

(3) Implementar el Algoritmo de Paso de Niño, Paso de Gigante y utilizarlo para calcular $\log_2(3)$ y $\log_3(2)$ en $\mathbb{Z}_{5^8}$ y en $\mathbb{Z}_{234457}$.

(4) Comprobar que la salida del la parte de cálculo del Algoritmo 24 igual a $\mathrm{Ind_g(y)}$.

(5) Consideremos el polinomio $f = x^5 + x + 4 \in \mathbb{F}_7[X]$ y sea a una raíz de $\mathbb{F}_7$ en alguna extensión de $\mathbb{F}_7$.

   (a) Demostrar que f es irreducible sobre $\mathbb{F}_7$.

   (b) Demostrar que a es un elemento primitivo de $\mathbb{F}_{7^5}$.

   (c) Calcular los índices en base a de los elementos de $\mathbb{F}_7^*$.

   (d) Sea $B = \{a + c : c \in \mathbb{F}_7\}$. Calcular los índices en base a de los elementos de B.

   (e) Calcular $\mathrm{Ind_a}(1 + \mathrm{a}^2)$ utilizando el Algoritmo 24.

Capítulo 6

Curvas Elípticas

En este capítulo vamos a introducir las curvas elípticas y a dotarlas de una estructura de grupo abeliano (de hecho de muchas estructuras de grupo abeliano). La prestación que las curvas elípticas aportan en criptografía es la de proporcionar una inagotable familia de grupos abelianos en los que utilizar los criptosistemas basados en la dificultad de resolver el Problema del Logaritmo Discreto.

6.1 Curvas afines y proyectivas

En esta sección K denota un cuerpo arbitrario. Identificamos el *plano afín* $\mathbb{A}^2(K)$ sobre K con los elementos de K^2. Una recta de $\mathbb{A}^2(K)$ es el conjunto de puntos $(x, y) \in K^2$ que satisfacen una ecuación del tipo

$$ax + by + c = 0$$

con $a, b, c \in K$ y $(a, b) \neq 0$. Sabemos que por cada dos puntos distintos de $\mathbb{A}^2(K)$ pasa una única recta y que dos rectas distintas de $\mathbb{A}^2(K)$ se intersectan en como mucho un punto. Sin embargo si c y d son elementos distintos de K entonces las rectas $ax+by+c = 0$ y $ax+by+d = 0$ son paralelas, es decir no se cortan en ningún punto. Para evitar esto se introduce el plano proyectivo.

Definimos en $K^3 \setminus \{(0, 0, 0)\}$ la siguiente relación de equivalencia:

$$a \sim b \quad \Leftrightarrow \quad a = tb \text{ para algún } t \in K \setminus \{0\}, \quad (a, b \in K^3 \setminus \{(0, 0, 0)\}).$$

Al conjunto cociente $\mathbb{P}^2(K) = (K^3 \setminus \{(0, 0, 0)\})/\sim$ lo llamamos *plano proyectivo*. Cada elemento de $\mathbb{P}^2(K)$ está representado por un elemento no nulo de K^3, de forma que dos elementos de K^3 representan el mismo elemento del plano proyectivo si son proporcionales. Se llaman *coordenadas homogéneas* de un punto del plano proyectivo a cualquier terna que lo represente. El punto del plano proyectivo representado por (x, y, z), o cualquier vector de la forma $\lambda(x, y, z)$ con $\lambda \neq 0$, lo denotamos por $[x, y, z]$ y decimos que $[x, y, z]$ son coordenadas homogéneas de dicho punto.

Podemos identificar los elementos de $\mathbb{P}^2(K)$ con los subespacios vectoriales K^3 de dimensión 1 (sin el $(0, 0, 0)$). Por definición las de $\mathbb{P}^2(K)$ son los subespacios vectoriales de K^3 de di-

mensión 2. Es decir una recta de $\mathbb{P}^2(K)$ está formada por los puntos de $\mathbb{P}^2(K)$ cuyas coordenadas homogéneas $[x, y, z]$ satisfacen una ecuación del tipo

$$aX + bY + cZ = 0$$

con $(a, b, c) \in K^3 \setminus \{(0, 0, 0)\}$. Es fácil ver que por cada dos puntos distintos de $\mathbb{P}^2(K)$ pasa una única recta de $\mathbb{P}^2(K)$ y que dos rectas distintas de $\mathbb{P}^2(K)$ se cortan en un único punto (es decir, en $\mathbb{P}^2(K)$ no hay rectas paralelas).

Podemos considerar el plano afín como un subconjunto del plano proyectivo identificando el punto $(x, y) \in \mathbb{A}^2(K)$ con el punto de coordenadas proyectivas $(x, y, 1)$. Con esta identificación, los puntos de $\mathbb{P}^2(K)$ que están en $\mathbb{A}^2(K)$ son los que tienen coordenadas homogéneas $[x, y, z]$, con $z \neq 0$. Tal punto se identifica con el punto del plano afín de coordenadas $\left(\frac{x}{z}, \frac{y}{z}\right)$. Los puntos de $\mathbb{P}^2(K)$ que no están en $\mathbb{A}^2(K)$ forman una recta, precisamente la recta de ecuación $Z = 0$. Esta recta se suele llamar *recta del infinito*. De esta forma podemos considerar el plano proyectivo como un plano afín al que se le ha añadido una recta, "en el infinito". Si L es una recta de $\mathbb{P}^2(K)$ distinta de la recta del infinito entonces viene dada por una ecuación $aX + bY + cZ = 0$ con $(a, b) \neq (0, 0)$. En tal caso $L \cap \mathbb{A}^2(K)$ es la recta afín de ecuación $ax + by + c = 0$ y L está formada por los puntos de esta recta afín y el punto $[-b, a, 0]$ de la recta del infinito. Por tanto podemos pensar que para pasar de $\mathbb{A}^2(K)$ hemos añadido una recta que tiene un punto para cada haz de rectas paralelas de $\mathbb{A}^2(K)$.

Una *curva algebraica* del plano afín $\mathbb{A}^2(K)$ es el conjunto de puntos del plano afín que satisfacen una ecuación de la forma $f(x, y) = 0$, donde f es un polinomio no nulo de $K[X, Y]$. Tal conjunto lo denotamos por $V(f)$. Está claro que

$$V(fg) = V(f) \cap V(g).$$

Un polinomio $F \in K[X, Y, Z]$ se dice que es homogéneo de grado d si es suma de monomios de grado d. Si F es un polinomio arbitrario en tres variables y $[x, y, z]$ y $[x_1, y_1, z_1]$ son coordenadas homogéneas del mismo punto del plano proyectivo entonces $F(x, y, z)$ y $F(x_1, y_1, z_1)$ tienen, en general valores distintos. Por tanto no tiene sentido evaluar los polinomios en tres variables en puntos del plano proyectivo. Sin embargo, como existe un escalar $\lambda \in K^*$ tal que $(x_1, y_1, z_1) = \lambda(x, y, z)$, si F es homogéneo de grado d entonces $F(x_1, y_1, z_1) = \lambda^d F(x, y, z)$ y por tanto $F(x_1, y_1, z_1) = 0$ si y solo si $F(x, y, z) = 0$. Esto muestra que la siguiente definición tiene sentido. Una *curva del plano proyectivo* es el conjunto formado por los puntos cuyas coordenadas homogéneas satisfacen una ecuación del tipo $F(X, Y, Z) = 0$, donde F es un polinomio homogéneo no nulo en $K[X, Y, Z]$. En tal caso decimos que el polinomio define la curva de ecuación $F(X) = 0$ y al conjunto de puntos que forman la curva lo denotamos $V(F)$.

A menudo abusaremos de la notación y diremos que una curva algebraica afín es un polinomio $f \in K[X, Y]$ y una curva proyectiva es un polinomio homogéneo no nulo F. De hecho dos polinomios proporcionales definen la misma curva y en consecuencia entenderemos que si $\lambda \in K^*$ entonces $f \in K[X, Y]$ y λf definen la misma curva afín y $F \in K[X, Y, Z]_d$ y λF definen la misma curva proyectiva. Sin embargo, hay que tomar esto con precaución pues, por ejemplo f y f^2, definen la misma curva. Más generalmente toda curva se puede definir por un polinomio que no es divisible por el cuadrado de ningún otro polinomio.

Obsérvese que los puntos afines de la curva de $\mathbb{P}^2(K)$ de ecuación $F(X,Y,Z)=0$ forman una curva algebraica del plano afín, en concreto la curva de ecuación $F(X,Y,1)=0$. El polinomio F° dado por

$$F^\circ(X,Y)=F(X,Y,1)$$

se llama *deshomogeneizado* de F. En otras palabras, cada curva proyectiva tiene asociada una curva afín, de forma que si F es el polinomio que define la curva proyectiva, entonces F° es el polinomio que define la correspondiente curva afín.

Recíprocamente, si $f=f_0+f_1+f_2+\cdots+f_d\in K[X,Y]$, donde f_i es homogéneo de grado i y $f_d\neq 0$, entonces se llama *homogeneizado* de f al polinomio

$$\widehat{f}=f_0Z^d+f_1Z^{d-1}+f_2Z^{d-2}+\cdots+f_{d-1}Z+f_d.$$

Por ejemplo, el homogeneizado del polinomio X^3-X^2+XY+Y es $X^3-X^2+XY+Y^2$.

Las siguientes propiedades son fáciles de verificar, donde las letras minúsculas representan elementos no nulos de $K[X,Y]$ y las mayúsculas polinomios homogéneos de $K[X,Y,Z]$.

(1) $\widehat{f_1f_2}=\widehat{f_1}\widehat{f_2}$.

(2) $(F_1F_2)^\circ=F_1^\circ F_2^\circ$.

(3) $\widehat{f^\circ}=f$.

(4) $F=Z^k\widehat{F^\circ}$, donde Z^k es la mayor potencia de Z que divide a F.

(5) Si $\widehat{f}$ es irreducible, entonces f es irreducible.

(6) Si f es irreducible, entonces $\widehat{f}$ es irreducible precisamente si $f(0,0)\neq 0$.

(7) Si F es irreducible entonces F° es irreducible.

(8) Si F° es irreducible, entonces F es irreducible precisamente si no es divisible por Z.

Un *cambio de coordenadas* en el plano afín es una aplicación del tipo

$$(x,y)\mapsto(a_{11}x+a_{12}y+b_1,a_{21}x+a_{22}y+b_2),\quad \text{con}\quad a_{11}a_{22}-a_{12}a_{21}\neq 0.$$

Matricialmente un cambio de coordenadas toma la forma de la siguiente biyección de $\mathbb{A}^2(K)$

$$X=\begin{pmatrix}x\\y\end{pmatrix}\mapsto AX+b$$

donde X representa un punto genérico del plano afín, b es otro punto fijo del plano afín fijo y $A\in \mathrm{GL}_2(K)$. Recuérdese que $\mathrm{GL}_n(K)$ representa la matrices invertibles con n filas y entradas en K.

Un *cambio de coordenadas homogéneas* u *homografía* del plano proyectivo es una aplicación del tipo

$$X\mapsto AX$$

donde X representa las coordenadas homogéneas de un punto del plano proyectivo y $A \in \mathrm{GL}_3(K)$.

Obsérvese que el cambio de coordenadas afín $X \to AX+b$ transforma los puntos de la curva algebraica de ecuación $f(X) = 0$ en los puntos de la curva algebraica de ecuación $f(A^{-1}(X - b)) = 0$. Análogamente el cambio de coordenadas homogéneas $\phi : X \to AX$ transforma la curva del plano proyectivo $F(X) = 0$ en la curva que tiene ecuación $F(A^{-1}X) = 0$. Está claro que las propiedades geométricas de las curvas algebraicas (respectivamente, las curvas del plano proyectivo) no varían al realizar un cambio de coordenadas afín (respectivamente, de coordenadas homogéneas). Por tanto, decimos que dos *curvas* son *equivalentes* si se puede pasar de una a otra mediante un cambio de coordenadas. Más precisamente, dos curvas C_1 y C_2 se dice que son equivalentes si existe un cambio de coordenadas afines ϕ (coordenadas homogéneas en el caso de curvas proyectivas) tal que $P \in C_1$ si y solo si $P \in C_2$.

Es tentador definir el grado de una curva del plano afín o proyectivo como el grado del polinomio que la representa. Está claro que este grado no variaría mediante un cambio de coordenadas afines (homogéneas). Sin embargo este concepto de grado es engañoso pues las ecuaciones $X = 0$ y $X^2 = 0$ definen las mismas curvas. Cada curva tiene una ecuación del tipo $F_1F_2 \dots F_n = 0$, con $F_1, \dots, F_n$ polinomios irreducibles no asociados (y homogéneos, para el caso proyectivo). Entonces definiremos el *grado* de la curva es el del polinomio $F = F_1F_2 \dots F_n$.

Una *curva de Weierstrass* es una cúbica del siguiente tipo

$$Y^2Z + a_1XYZ + a_3Y^2Z = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3. \tag{6.1}$$

Abusaremos de la notación e identificaremos la curva de Weierstrass con la *Ecuación de Weierstass*, en su versión proyectiva dada por (6.3) o por la *curva de Weierstrass afín* asociada:

$$y^2 + a_1xy + a_3y^2 = x^3 + a_2x^2 + a_4x + a_6. \tag{6.2}$$

Más adelante, veremos el porqué de esta extraña numeración de los coeficientes.

El único punto de intersección de una curva de Weierstass con la recta del infinito es $[0, 1, 0]$.

Supongamos que x_1, x_2, x_3 y sean $P_1, P_2, P_3 \in \mathbb{P}^2(K)$ de forma que $P_i = [x_i]$. Está claro que P_1, P_2 y P_3 están alineados en $\mathbb{P}^2(K)$ si y solo si x_1, x_2, x_3 son linealmente dependientes en K^3. Un *sistema de referencia del plano proyectivo* es una lista formada por cuatro puntos del plano proyectivo de forma que no haya tres alineados.

Proposición 6.1 *Si (P_1, P_2, P_3, P_4) y (Q_1, Q_2, Q_3, Q_4) son dos sistemas de referencia del plano proyectivo entonces existe una única homografía ϕ del plano proyectivo tal que $\phi(P_i) = Q_i$ para todo i.*

Demostración. Podemos suponer sin pérdida de generalidad que el primer sistema de referencia está formado por $P_1 = [1, 0, 0]$, $P_2 = [0, 1, 0]$, $P_3 = [0, 0, 1]$ y $P_4 = [1, 1, 1]$. Si el otro sistema de referencia está formado por puntos Q_i coordenadas homogéneas $[x_i]$ con $x_i \in K^3$, entonces la columna i-ésima de la matriz que define una homografía ϕ tal que $\phi(P_i) = Q_i$ tienen que ser de la forma $\lambda_i x_i$ con $\lambda_i \neq 0$. Eso garantiza que $\phi(e_i) = P_i$ para $i = 1, 2, 3$. Para que además $\phi(e_4) = P_4$, necesitamos que $\lambda x_4 = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$ para algún $\lambda \neq 0$. Como x_1, x_2 y x_3 son linealmente independientes, $\left(\frac{\lambda_1}{\lambda}, \frac{\lambda_2}{\lambda}, \frac{\lambda_3}{\lambda}\right)$ son las coordenadas de x_4 en la base

(x_1, x_2, x_3). Por tanto, la matriz que define ϕ está únicamente determinada salvo una constante pero eso implica que la homografía está únicamente determinada. Más concretamente si las coordenadas de x_4 en la base (x_1, x_2, x_3) son $\alpha_1, \alpha_2, \alpha_3$ la homografía viene dada por cualquier matriz cuya i-ésima columna sea $\alpha\alpha_i x_i$ para un $\alpha \neq 0$, el mismo para las tres columnas. Como dos matrices determinan la misma homografía si y solo si son proporcionales, esto acaba la demostración. ∎

6.2 Teorema de Bezout

En esta sección D es un dominio de factorización única (DFU). Dados dos polinomios

$$f = f_0 + f_1X + \cdots + f_mX^m, \quad g = g_0 + g_1X + \cdots + g_nX^n$$

de grados m y n respectivamente, llamamos *resultante* de f y g al determinante $R(f, g)$ de la siguiente matriz cuadrada con mn filas:

$$[R(f,g)] = \begin{pmatrix} f_0 & f_1 & \cdots & \cdots & f_m & & & \\ & f_0 & f_1 & \cdots & \cdots & f_m & & \\ & & \ddots & \ddots & & & \ddots & \\ & & & f_0 & f_1 & \cdots & \cdots & f_m \\ g_0 & g_1 & \cdots & \cdots & g_n & & & \\ & g_0 & g_1 & \cdots & \cdots & g_n & & \\ & & \ddots & \ddots & & & \ddots & \\ & & & g_0 & g_1 & \cdots & \cdots & g_n \end{pmatrix}$$

formada con m filas con los coeficientes de f_i y n filas con los coeficientes g_i.

Proposición 6.2 *Las siguientes condiciones son equivalentes para $f, g \in D[X] \setminus \{0\}$.*

(1) f y g tienen un divisor común en $D[X]$ de grado > 0.

(2) Existen $g_1, f_1 \in D[X] \setminus \{0\}$ tales que $g_1 f + f_1 g = 0$, $\deg(g_1) < \deg(g)$ y $\deg(f_1) < \deg(f)$.

(3) $R(f, g) = 0$.

Demostración. (1) implica (2). Si h es un divisor común de f y g entonces $f_1 = \frac{f}{h}$ y $g_1 = -\frac{g}{h}$ satisfacen (2).

(2) implica (1). Si $g_1 f + f_1 g = 0$ y f y g no tienen factores comunes entonces g divide a g_1, en contra de que $\deg(g) < \deg(g_1)$.

(2) si y solo si (3). Supongamos que $m = \deg(f)$ y $n = \deg(g)$ y consideremos dos polinomios

$$a = a_0 + a_1X + \cdots + a_{n-1}X^{n-1} \quad \text{y} \quad b = b_0 + b_1X + \cdots + b_{m-1}X^{m-1}$$

con coeficientes en el cuerpo de fracciones de D y multipliquemos

$$(a_0, a_1, \ldots, a_{n-1}, b_0, b_1, \ldots, b_{m-1})[R(f,g)] = (c_0, \ldots, c_{m+n-1}).$$

Entonces

$$af + bg = c = c_0 + c_1 X + \cdots + c_{m+n-1} X^{m+n-1}.$$

Esto implica que (2) se verifica si y solo si la matriz $[R(f,g)]$ no es invertible o sea si y solo si $R(f,g) = 0$. ∎

Si el cuerpo es algebraícamente cerrado entonces la propiedad de que dos rectas distintas de $\mathbb{P}^2(K)$ se intersequen en exactamente un punto se extiende a curvas proyectivas arbitrarias:

Teorema 6.3 (Teorema de Bezout) *Sean $F \in K[X,Y,Z]_m$ y $G \in K[X,Y,Z]_n$.*

(1) Si $mn > 0$ entonces $F(\overline{K}) \cap G(\overline{K}) \neq \emptyset$.

(2) Si $|F(\overline{K}) \cap G(\overline{K})| > mn$ entonces F y G tienen algún divisor común.

Demostración. (1) Escribimos

$$F = \sum_{i=0}^{m} f_{m-i} Z^i \quad \text{y} \quad G = \sum_{i=0}^{n} g_{n-i} Z^i$$

con $f_i, g_i \in K[X,Y]_i$ y utilizamos esto para construir $R(F,G)$ mirando F y G como polinomios en Z con coeficientes en $K[X,Y]$. Es fácil ver que $R(F,G)$ es un polinomio homogéneo de grado mn. Por tanto tiene $R(F,G)$ tiene algún cero (x_0, y_0). Por la Proposición 6.2, $F(x_0, y_0, Z)$ y $G(x_0, y_0, Z)$ tienen un divisor común $H(Z)$ de grado > 0. Aplicando de nuevo que F es algebraicamente cerrado deducimos que $H(Z)$ tiene una raíz z_0. Eso implica que $(x_0, y_0, z_0) \in F(\overline{K}) \cap G(\overline{K})$.

(2) Supongamos que $F(\overline{K}) \cap G(\overline{K})$ tiene $mn+1$ puntos distintos (x_i, y_i, z_i) $(i = 0, 1, \ldots, mn)$; tomemos todas las rectas definidas por cada dos de esos puntos y elijamos un punto que no pertenezca a ninguna de ellas. Después de hacer un cambio de coordenadas podemos suponer que dicho punto es $[0,0,1]$. Eso implica que $(x_i, y_i) \neq 0$ para todo $i = 0, 1, \ldots, mn$. Como $F(x_i, y_i, z_i) = G(x_i, y_i, z_i) = 0$, se tiene que $Z - z_i$ es un factor común de $F(x_i, y_i, Z)$ y $G(x_i, y_i, Z)$. Por la Proposición 6.2, $R(F(x_i, y_i, Z), G(x_i, y_i, Z)) = 0$. Por tanto $R(F,G)$ es un polinomio homogéneo de grado mn que se anula en todos los puntos de cada una de las rectas $L_i = y_i X - x_i Y$. Eso implica que cada una de estas rectas divide a $R(F,G)$. Si $L_i = L_j$ con $i \neq j$ entonces $[x_i, y_i, z_i]$ y $[x_j, y_j, z_j]$ son puntos distintos de L_i y por tanto L_i es la recta que pasa por esos dos puntos, pero por la construcción obtenemos una contradicción pues $[0,0,1]$ también está en esta recta. Luego todas las rectas L_i son distintas y, por tanto, el producto de ellas divide a $R(F,G)$, pero como $R(F,G)$ tiene grado mn, solo puede ser múltiplo de un producto de $mn+1$ rectas si es 0. Luego $R(F,G) = 0$. Aplicando una vez más la Proposición 6.2 a los polinomios F y G vistos como elementos de $K[X,Y][Z]$ deducimos que , con lo que $F(X,Y,Z)$ y $G(X,Y,Z)$ tienen un factor en $\overline{K}[X,Y,Z]$ y por las propiedades de factorización única también tienen un divisor común en $K[X,Y,Z]$. ∎

El siguiente corolario es consecuencia inmediata del Teorema 6.3.

Corolario 6.4 *Supongamos que $F \in K[X,Y,Z]_d$ y L es una recta. Si $V(F) \cap V(L)$ tiene más de d puntos entonces L divide a F.*

6.3 Puntos singulares y rectas tangentes

Consideremos la curva afín dada por el polinomio $f \in K[x,y]$ y sea $P = (a,b) \in V(F)$. Entonces la *recta tangente* a $V(f)$ en (a,b) es por definición la recta de ecuación

$$f'_x(P)(x-a) + f'_y(P)(y-b) = 0$$

si es que esto tiene sentido. Es decir, si $(f'_x(P), f'_y(P)) \neq (0,0)$. En tal caso decimos que P es un *punto regular* de $V(f)$, en caso contrario, es decir si $(f'_x(P), f'_y(P)) = (0,0)$, entonces decimos que P es un *punto singular*.

De forma similar si F es un polinomio homogéneo de $K[X,Y,Z]$ y $P \in V(F)$, entonces decimos que $P \in \mathbb{P}^2$ es un *punto singular* de $V(F)$ si $F'_X(P) = F'_Y(P) = F'_Z(P) = 0$ y en caso contrario decimos que P es un *punto regular* de $V(F)$. En este segundo caso, si $P = [a,b,c]$ entonces la recta tangente a F en P es la que tiene la siguiente ecuación:

$$F'_X(P)X + F'_Y(P)Y + F'_Z(P)Z = 0.$$

Denotaremos la recta tangente a F en P como $T_P(F)$.

Decimos que F define una *curva no singular* si la curva definida por F en el plano proyectivo de una clausura algebraica de K no tiene ningún punto singular.

Proposición 6.5 *Sean F y G polinomios homogéneos de $K[X,Y,Z]$ y sea $P \in V(F) \cap V(G)$. Entonces P es un punto singular de $V(FG)$.*

Demostración. Como $F(P) = G(P) = 0$ tenemos $(FG)'_X(P) = F'_X(P)G(P) + F(P)G'_X(P) = 0$ y análogamente $(FG)'_Y(P) = (FG)'_Z(P) = 0$. ∎

Corolario 6.6 *Si F define una curva no singular entonces F es irreducible.*

Proposición 6.7 *Sean ϕ es una homografía de $\mathbb{P}^2$, F es un polinomio homogéneo y $P \in \mathbb{P}^2$. Entonces*

(1) $F^\phi(X,Y,Z) = F(\phi^{-1}(X,Y,Z))$ es un polinomio de homogéneo del mismo grado que F.

(2) $P \in V(F)$ si y solo si $\phi(P) \in V(F^\phi)$.

(3) P es un punto singular de $V(F)$ si y solo si $\phi(P)$ es un punto singular de $V(F^\phi)$. En tal caso $T_{\phi(P)}(F^\phi) = T_P(F)^\phi$.

Sea F un polinomio homogeneo de grado d y sea $P \in V(F)$. De la Proposición 6.1 sabemos que existe una homografía ϕ tal que $\phi(P) = [0,0,1]$. En tal caso decimos que ϕ

es una homografía *centrada* en P. Entonces $(0,0)$ es un punto de la curva afín $F^\phi(x,y) = (F \circ \phi^{-1})(x,y,1)$ pues

$$F^\phi(0,0) = F(\phi^{-1}(0,0,1)) = F(P) = 0.$$

Luego

$$G = F^\phi = f_1 Z^{d-1} + f_2 Z^{d-2} + \cdots + f_{d-1} Z + f_d$$

donde f_i es un polinomio homogéneo de grado i. A grandes rasgos: cuanto mayor sea el menor k tal que $f_k \neq 0$, mayor será "la multiplicidad" de P como cero de F.

Obsérvese que $f_i(0,0) = 0$ para todo i con lo que $G'_Z(0,0,1) = 0$. Además, si $i \geq 2$ entonces $(f'_i)_X(0,0) = (f'_i)_Y(0,0)$. Por tanto si $f_1 = aX + bY$ entonces $a = G'_X(0,0,1)$ y $b = G'_Y(0,0,1)$. Eso implica que P es singular en F si y solo si $f_1 = 0$ y si P es regular F entonces $T_P(F)^\phi$ es la recta de ecuación $f_1(X,Y,Z) = 0$ con lo que $T_P(F) = f_1^{\phi^{-1}}$.

6.4 Orden de intersección

Sean $F \in K[X,Y,Z]_d$ y $L \in K[X,Y,Z]_1$ y sea $P \in V(F) \cap V(L)$. Fijamos una homografía ϕ centrada en P y pongamos

$$\begin{aligned} f &= F^\phi(x,y,1) = f_1 + \cdots + f_d \\ l &= L^\phi(x,y,1) = bx - ay \end{aligned}$$

con $(a,b) \neq (0,0)$. Entonces $\alpha(t) = (at, bt) (= [at, bt, 1])$ es una parametrización de $L^\phi(K) \cap \mathbb{A}^2(K)$ y $f(\alpha(t))$ es un polinomio en t, que tiene 0 como raíz. Más precisamente

$$f(\alpha(t)) = t f_1(a,b) + t^2 f_2(a,b) + \cdots + t^d f_d(a,b).$$

Lema 6.8 *$f(\alpha(t))$ es el polinomio nulo si y solo si L divide a F.*

Demostración. Si L divide a F, entonces $L(\overline{K}) \subseteq F(\overline{K})$. Eso implica que $f(\alpha(t)) = 0$ para todo $t \in \overline{K}$. Como $\overline{K}$ es infinito $f(\alpha(t)) = 0$.

Recíprocamente, si $f(\alpha(t)) = 0$ entonces $f_r(a,b) = 0$ para todo $r = 1, \ldots, d$. Pongamos $f_r = \sum_{i=0}^r c_i X^i Y^{r-i}$. Podemos suponer sin pérdida de generalidad que $b \neq 0$. Entonces

$$0 = f_r(a,b) = \sum_{i=0}^r c_i a^i b^{r-i} = b^r \sum_{i=0}^r c_i \left(\frac{a}{b}\right)^i.$$

Por tanto, $X - \frac{a}{b}$ divide a $b^r f_r(X,1)$. Pongamos $b^r f_r(X,1) = \left(X - \frac{a}{b}\right) u(X)$ con u de grado $r-1$. Entonces

$$b^r f_r(X,Y) = b^r Y^r f_r\left(\frac{X}{Y}, 1\right) = Y^r \left(\frac{X}{Y} - \frac{a}{b}\right) u\left(\frac{X}{Y}\right) = b^{-1} l(X,Y) Y^{r-1} u\left(\frac{X}{Y}\right).$$

Como u tiene grado $r-1$ $Y^{-1} u\left(\frac{X}{Y}\right) \in K[X,Y]$ Esto demuestra que $l(X,Y) = L^\phi(X,Y,1)$ divide a $f = F^\phi(X,Y,1)$. Para acabar la demostración basta aplicar el Problema 6.1.1. ∎

Llamaremos *orden de intersección* de F y L en $P \in \mathbb{P}^2(K)$ al siguiente número.

$$I(P, L \cap F) = \begin{cases} 0, & \text{si } P \notin V(L) \cap V(F); \\ \infty, & \text{si } P \in V(L) \text{ y } L|F; \\ \text{multiplicidad de 0 como raíz de } f(\alpha(t)), & \text{en otro caso.} \end{cases}$$

El lector debería verificar que esta definición no depende de la homografía ϕ (Problema 6.4.5).

Por ejemplo, si P es un punto no singular de F y $L_T = T_P(F)$ entonces $l_T(x, y) = L_T^\phi(x, y, 1) = f_1(x, y) = Ax + By$ con $(A, B) \in K^2 \setminus \{(0, 0)\}$ y tenemos que $I(P, L \cap F) = 1$ si y solo si $f_1(a, b) \neq 0$ si y solo si $Aa + Bb \neq 0$ si y solo si (A, B) y $(b, -a)$ no son proporcionales, si y solo si $L \neq L_T$. Por tanto,

Proposición 6.9 *Si P es un punto no singular de F entonces $I(P, L \cap F) > 1$ si y solo si L es la recta tangente a F en P.*

Proposición 6.10 *Sean $P = [x_0, y_0, z_0] \in V(L) \cap V(F)$ y $P \neq P_1 = [x_1, y_1, z_1] \in V(L)$. Entonces $I(P, L \cap F)$ es la multiplicidad de 0 como raíz del polinomio $g(t) = F(x_0 + tx_1, y_0 + ty_1, z_0 + tz_1)$.*

Demostración. Elegimos un tercer punto $P_2 = [x_2, y_2, z_2] \notin V(L)$. Eso implica que P, P_1 y P_2 no están alineados, con lo que existe una homografía ϕ tal que $\phi(P_2), \phi(P), \phi(P_1)$ es la base canónica. (Más rigurosamente $\phi(x_2, y_2, z_2) = (1, 0, 0)$, $\phi(x_1, y_1, z_1) = (0, 1, 0)$ y $\phi(x_0, y_0, z_0) = (0, 0, 1)$, Luego $\Phi^{-1}[X, Y, Z] = [X(x_2, y_2, z_2) + Y(x_1, y_1, z_1) + Z(x_0, y_0, z_0)]$. Por tanto,

$$l(x, y) = (L^\phi)(x, y, 1) = xL(x_2, y_2, z_2) + yL(x_1, y_1, z_1) + L(x_0, y_0, z_0) = xL(x_2, y_2, z_2).$$

Luego, una parametrización de $l(x, y) = 0$ es $\alpha(t) = (0, t)$. Entonces $f(\alpha(t)) = F^\phi(0, t, 1) = F(x_0 + tx_1, y_0 + ty_1, z_0 + tz_1)$. ∎

Corolario 6.11 *Si $F \in K[X, Y, Z]_d$ y L es una recta tal que $\sum_{P \in \mathbb{P}^2(K)} I(P, L \cap F) > d$ entonces L divide a F.*

Demostración. Podemos suponer sin pérdida de generalidad que K es algebraicamente cerrado y en particular es infinito. Supongamos que L no divide a F. Por el Corolario 6.4 deducimos que $V(L) \cap V(F)$ es finito. Después de hacer una homografía podemos suponer que L es la recta del infinito y que ningún punto de $V(L) \cap V(F)$ está en la recta $Y = 0$. Luego los puntos de $V(L) \cap V(F)$ son de la forma $[x, 1, 0]$. Por tanto, si ponemos $g(X) = F(X, 1, 0)$ entonces

$$V(L) \cap V(F) = \{[r, 1, 0] : r \text{ es raíz de } g(X)\}.$$

Sea $h(X) = g(r + X)$. Entonces $h(t) = F(r + t, 1, 0) = F((r, 1, 0) + t(1, 0, 0))$. Por la Proposición 6.10, $I([r, 1, 0], L \cap F)$ coincide con la multiplicidad de 0 como raíz de h y esta coincide con la multiplicidad de r como raíz de g. Por tanto, $\sum_{P \in \mathbb{P}^2(K)} I(P, L \cap F)$ es la suma de las multiplicidades de las raíces de $g(X)$, que es menor o igual que el grado de g que a su vez es menor o igual que d. ∎

Un *punto de inflexión* de F es un punto no singular P de F tal que $I(P, T_P(F) \cap F) > 2$. Una *curva elíptica* sobre K es una curva proyectiva no singular $F \in K[X, Y, Z]_3$ con un punto de inflexión en $F(\overline{K})$.

Una *curva de Weierstrass* es una cúbica del siguiente tipo

$$Y^2Z + a_1XYZ + a_3Y^2Z = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3. \tag{6.3}$$

Abusaremos de la notación e identificaremos la curva de Weierstrass con la *Ecuación de Weierstass*, en su versión proyectiva dada por (6.3) o por la *curva de Weierstrass afín* asociada:

$$y^2 + a_1xy + a_3y^2 = x^3 + a_2x^2 + a_4x + a_6. \tag{6.4}$$

Más adelante, veremos el porqué de esta extraña numeración de los coeficientes.

Proposición 6.12 *Una cúbica no singular F definida sobre K tiene un punto de inflexión en $V(F)$ si y solo si es equivalente sobre K a una curva de Weierstrass.*

Demostración. En primer lugar observemos que $[0, 1, 0]$ es un punto de inflexión de cualquier Curva de Weierstrass (Problema 7). Para demostrar el recíproco consideramos la forma general de una cúbica:

$$F = \sum_{i+j+k=3} a_{X^iY^jZ^k} X^iY^jZ^k.$$

Después de transformar F con una homografía ϕ tal que $\phi(P) = [0, 1, 0]$, podemos suponer que $[0, 1, 0]$ es un punto de inflexión y que la recta tangente a este punto es la recta del infinito. Eso tiene las siguientes consecuencias:

(1) $\underline{a_{Y^3} = 0}$: Esto es consecuencia de que $F(0, 1, 0) = 0$.

(2) $\underline{a_{Y^2} \neq 0 \text{ ó } a_{XY^2} \neq 0}$: Sea ϕ la homografía centrada en $[0, 1, 0]$ dada por $\phi(x, y, z) = (z, x, y)$. Entonces

$$F^\phi(X, Y, 1) = a_{Y^2}X + a_{XY^2}Y + \text{ Términos de grado superior.}$$

Por tanto, $f_1(X, Y) = a_{Y^2}X + a_{XY^2}Y$. Como $[0, 1, 0]$ es un punto no singular $f_1 \neq 0$ y por tanto $a_{Y^2} \neq 0$ ó $a_{XY^2} \neq 0$.

(3) $\underline{a_{XY^2} = 0 \text{ y } a_{Y^2} \neq 0}$: Como la recta tangente a F en $[0, 1, 0]$ es la recta del infinito deducimos que $Z = T_{[0,1,0]}(F) = f_1(\phi(X, Y, Z)) = f_1(Z, X, Y) = a_{Y^2}Z + a_{XY^2}X$. De donde, efectivamente se deduce que $a_{XY^2} = 0$ y por (2), $a_{Y^2} \neq 0$.

(4) $\underline{a_{X^2Y} = 0}$. Para ver esto utilizamos que $[0, 1, 0]$ es un punto de inflexión y $L_T = T_P(F) = Z$. Calculamos,

$$\begin{aligned} F^\phi(X, Y, 1) &= F(Y, 1, X) \\ &= a_{Y^2}X + a_{X^2Y}Y^2 + a_{XY}XY + a_{Y^2}X^2 + \\ &\quad a_{X^3}Y^3 + a_{X^2}Y^2X + a_{X^2Y}YX^2 + a_3X^3, \\ L_T^\phi(X, Y, 1) &= L_T(Y, 1, X) = X. \end{aligned}$$

y elegimos la parametrización $\alpha(t) = (0, t)$ de L_T^ϕ. Como

$$F^\phi(\alpha(t)) = a_{X^2Y}t^2 + a_{X^3}t^3$$

y $[0, 1, 0]$ un punto de inflexión se ha de cumplir $a_{X^2Y} = 0$.

(5) $\underline{a_{X^3} \neq 0}$. Como además F es no singular, Z no divide a F y eso implica que $a_{X^3} \neq 0$.

En resumen, tenemos $a_{Y^3} = a_{XY^2} = a_{X^2Y} = 0$ y $a_{Y^2} \neq 0 \neq a_{X^3}$. Para conseguir una Ecuación de Weierstrass solo faltaría que $-a_{Y^2} = a_{X^3} = 1$. De hecho, como la curva viene determinada también por λF, para todo $\lambda \in K^*$. Bastaría con que $-a_{Y^2} = a_{X^3}$. Para ello vamos a hacer un cambio de coordenadas, considerando una homografía del tipo $\phi(x, y, z) = (tx, ty, z)$, para t un elemento no nulo de K a especificar. Este homografía deja $[0, 1, 0]$ y la recta del infinito invariantes, con lo que todo lo dicho arriba permanece válido. Entonces

$$F^\phi = \sum_{i+j+k=3} t^{-(i+j)} a_{X^iY^jZ^k} X^iY^jZ^k = \sum_{i+j+k=3} c_{X^iY^jZ^k} X^iY^jZ^k.$$

Deseamos conseguir $c_{X^3} = -c_{Y^2}$, o lo que es lo mismo $t^2 a_{X^3} = -t^3 a_{Y^2}$, lo cual se puede obtener eligiendo $t = -\frac{a_{X^3}}{a_{Y^2}}$. ∎

Se puede demostrar que si la característica de K es diferente de 2 entonces toda cúbica no singular sobre K es una curva elíptica. Además, por la Proposición 6.12, F es equivalente sobre $\overline{K}$ a una Curva de Weierstrass. En tal caso $[0, 1, 0]$ es un punto de inflexión cuya recta tangente es la recta del infinito. Esto nos garantiza que toda Ecuación de Weierstrass define una cúbica con un punto de inflexión, pero para que sea una curva elíptica además la curva habrá de ser no singular. Esto nos plantea el problema de decidir cuándo una Ecuación de Weierstrass, en la versión proyectiva (6.3) o afín (6.4), define una curva elíptica (o sea una curva no singular).

6.5 El producto cuerda-tangente

Proposición 6.13 *Sea F una cúbica no singular sobre K y L una recta. Entonces*

$$\sum_{P \in \mathbb{P}^2(K)} I(P, L \cap F) = 0, 1 \text{ ó } 3.$$

Demostración. En primer lugar observamos que F es irreducible por ser no singular (Corolario 6.6). Eso implica que L no divide a F y por tanto $I(P, L \cap F) < \infty$ para todo $P \in \mathbb{P}^2(K)$. Podemos suponer que $\sum_{P \in \mathbb{P}^2(K)} I(P, L \cap F) \neq 0$, o lo que es lo mismo $V(L) \cap V(F)$ contiene al menos un punto. Después de un cambio de coordenadas podemos suponer que dicho punto es $P_0 = [0, 1, 0]$ y que la recta tangente a F en dicho punto es la recta del infinito.

Consideramos la forma general de una cúbica:

$$F = \sum_{i+j+k=3} a_{X^iY^jZ^k} X^iY^jZ^k.$$

De $F(0,1,0)=0$ deducimos que $a_{Y^3}=0$. Sea ϕ la homografía centrada en $[0,1,0]$ dada por $\phi(x,y,z)=(z,x,y)$. Entonces

$$F^\phi(X,Y,1)=a_{Y^2}X+a_{XY^2}Y+\text{ Términos de grado superior.}$$

Por tanto, $f_1(X,Y)=a_{Y^2}X+a_{XY^2}Y$. Como la recta tangente a F en $[0,1,0]$ es la recta del infinito deducimos que $Z=T_{[0,1,0]}(F)=f_1(\phi(X,Y,Z))=f_1(Z,X,Y)=a_{Y^2}Z+a_{XY^2}X$. De donde se deduce que $a_{XY^2}=0$ y por (2), $a_{Y^2}\neq 0$. Resumiendo:

$$F=a_{Y^2Z}Y^2Z+a_{XYZ}XYZ+a_{X^2Y}X^2Y+a_{YZ^2}YZ^2+a_{X^3}X^3+a_{X^2Z}X^2Z+a_{XZ^2}XZ^2+a_{Z^3}Z^3.$$

con $a_{Y^2Z}\neq 0$. Además tenemos $F(X,Y,0)=X^2(a_{X^2Y}Y+a_{X^3}X)$.

Consideramos dos casos.

(1) Supongamos primero que $L=Z$, la recta del infinito. Como L no divide a F, $a_{X^2Y}\neq 0$ ó $a_{X^3}\neq 0$.

Elegimos la parametrización $\alpha(t)=(0,t)$ de L_T^ϕ. Como $a_{X^2Y}=0$ tenemos

$$F^\phi(\alpha(t))=a_{X^2Y}t^2+a_{X^3}t^3.$$

Si $a_{X^2Y}=0$ entonces $I([0,1,0],L\cap F)=3$, es decir $[0,1,0]$ un punto de inflexión. Con esto ya tenemos que $3\leq\sum_{P\in\mathbb{P}^2(K)}I(P,L\cap F)$ y combinándolo con el Corolario 6.11 deducimos que $\sum_{P\in\mathbb{P}^2(K)}I(P,L\cap F)=3$. En caso contrario, o sea si $a_{X^2Y}\neq 0$, tenemos que $I(P_0,L\cap F)=2$. Además el punto $Q=[1,t_1=-\frac{a_{X^3}}{a_{X^2Y}},0]$ pertenece a $V(L)\cap V(F)$ con lo que $I(Q,L\cap F)\geq 1$. Por tanto $3\leq\sum_{P\in\mathbb{P}^2(K)}I(P,L\cap F)$ con lo que de nuevo deducimos que $\sum_{P\in\mathbb{P}^2(K)}I(P,L\cap F)=3$.

(2) Supongamos que L no es la recta del infinito. Si $V(L)\cap V(F)=\{P_0\}$, entonces $\sum_{P\in\mathbb{P}^2(K)}I(P,L\cap F)=I(P_0,L\cap F)=1$, por la Proposición 6.9. En caso contrario existe $P_1\in V(L)\cap V(F)\setminus\{P_0\}$. Como L no es la recta del infinito, tenemos $V(L)\cap V(Z)=\{P_0\}$, con lo que podemos escribir $P_1=(x_0,y_0,1)$. Usamos la homografía con matriz

$$\begin{pmatrix}1&0&-x_0\\0&1&-y_0\\0&0&1\end{pmatrix}.$$

Esta homografía mantiene el punto P_0 y la recta del infinito y transforma P_1 en $[0,0,1]$ y la recta L en la recta X. Por tanto, podemos suponer que $P_1=[0,0,1]$ y $L=X$. Esto nos da la siguiente parametrización de la parte afín de L: $\alpha(t)=(0,t)$. Sustituyendo en la ecuación de F tenemos $F(\alpha(t))=F(0,t,1)=t(a_{YZ^2}+a_{Y^2Z}t)$. Si $a_{YZ^2}=0$ entonces $I(P_1,L\cap F)=2$. En caso contrario, $P_2=(0,-\frac{a_{YZ^2}}{a_{Y^2Z}},1)\in L\cap F$. En ambos casos $3\leq\sum_{P\in\mathbb{P}^2(K)}I(P,L\cap F)$ con lo que de nuevo deducimos que $\sum_{P\in\mathbb{P}^2(K)}I(P,L\cap F)=3$. ∎

Sean F una cúbica no singular y sean P y Q dos puntos de $V(F)$, no necesariamente distintos. Si $P\neq Q$, sea L la recta que pasa por P y Q. En caso contrario sea $L=T_P(F)$. Entonces $\sum_{R\in\{P,Q\}}I(R,L,F)\geq 2$, con lo que $\sum_{R\in\mathbb{P}^2(K)}I(R,L,F)=3$, por la Proposición 6.13. El *producto cuerda-tangente* de P y Q es el punto PQ que completa esta suma. Más precisamente,

PQ es el único punto R que cumple las siguientes propiedades:

$$PQ = \begin{cases} P, & \text{si } I(P, L \cap F) = 3 \text{ ó } P \neq Q \text{ y } I(P, L \cap F) = 2; \\ Q, & \text{si } I(Q, L, F) = 3 \text{ ó } P \neq Q \text{ y } I(Q, L, F) = 2; \\ R, & \text{si } \sum_{T \in \{P,Q\}} I(T, L, F) = 2 \text{ y } \sum_{T \in \{P,Q,R\}} I(T, L, F) = 3. \end{cases}$$

Los orígenes del producto-secante tangente se pueden ya encontrar en la *Aritmética* de Diofanto. Esta obra es una colección de problemas cuya solución está formada por números racionales que sean solución de una o más ecuaciones polinómicas con coeficientes racionales.

Ejemplo 6.14 Un ejemplo típico de Diofanto es el siguiente: Dividir 6 en dos partes de forma que su producto sea el cubo menos la raíz. En la terminología de Diofanto, la raíz es la incognita y dividir un número significa expresarlo como suma de dos números. Por tanto tenemos tres incógnitas, la "raíz" x y las partes y y z en las que "se divide" 6, de forma que $6 = y + z$ e $yz = x^3 - x$, o lo que es lo mismo $y(6 - y) = x^3 - x$. Otra parte de la terminología de Diofanto es que las soluciones buscadas son números racionales. Tres soluciones obvias son $(0, 0), (1, 0)$ y $(-1, 0)$. Estas soluciones están alineadas, por lo que no podemos obtener nuevas soluciones con la recta que pasa por dos de estos puntos. Salvo que se use uno de ellos "dos veces". Por ejemplo, La tangente a la curva $f = x^3 - x + y^2 - 6y$ en el punto $(1, 0)$ tiene ecuación $x = 1 + 3y$, sustituyendo x por $1 + 3y$ en la ecuación de la curva tenemos $0 = (1 + 3y)^3 - (1 + 3y) + y^2 - 6y = y^2(28 + 27y)$. Por tanto, el "tercer punto" se obtiene para $y = -\frac{28}{27}$ y $x = 1 - \frac{28}{9} = -\frac{19}{9}$.

Ejemplo 6.15 Supongamos que tenemos un cuadrado completamente lleno de pelotas alineadas. Es decir el número de total de pelotas es n^2. ¿Podremos construir con las n^2 pelotas una pirámide de forma que en el piso superior haya 1 pelota, en el segundo 4, en el tercero 9, y así sucesivamente? Claramente, sí es posible si $n = 1$, pero no es posible si $n = 2$, ni 3, ni 4. ¿Hay algún número n para el que podríamos construir una pirámide?

Si hubiera m pisos se tendría que verificar

$$1 + 2^2 + 3^2 + \cdots + m^2 = n^2.$$

Utilizando la igualdad

$$1 + 2^2 + 3^2 + \cdots + m^2 = \frac{m(m + 1)(2m + 1)}{6}$$

obtendríamos que (m, n) es una solución de la siguiente ecuación

$$y^2 = \frac{x(x + 1)(2x + 1)}{6}.$$

Por ejemplo, $P = (0, 0)$ y $Q = (1, 1)$ serían dos soluciones de esta ecuación. Apliquemos el método cuerda-tangente para obtener otra solución. La recta L que pasa por P y Q tiene ecuación $x = y$. Para obtener la intersección de L con la curva $F = y^2 - \frac{x(x+1)(2x+1)}{6}$ sustituimos y por x en la ecuación de F, lo que nos proporciona la siguiente ecuación:

$$x^2 = \frac{x(x + 1)(2x + 1)}{6},$$

que es equivalente a

$$x^3 - \frac{3}{2}x^2 + \frac{1}{2}x = 0.$$

Dos de las raíces son 0 y 1 (como era de esperar) y la tercera raíz a debe de verificar

$$0 + 1 + a = \frac{3}{2}.$$

Por tanto $a = \frac{1}{2}$. Esto nos proporciona un tercer punto $(\frac{1}{2}, \frac{1}{2})$ de F. Claro que este punto no resulta útil para nuestro problema pues no consideramos la opción de colocar un cuarto de pelota en una pirámide con medio piso. Observamos que si (x, y) es un punto de F, entonces $(x, -y)$ es otro punto, con lo que $(1, -1)$ y $(\frac{1}{2}, -\frac{1}{2})$ son otros dos puntos de la curva. Ya tenemos cinco puntos con coordenadas racionales. Calculemos ahora el producto secante tangente con $(1, 1)$ y $(\frac{1}{2}, -\frac{1}{2})$. La recta que pasa por estos dos puntos tiene ecuación $y = 3x - 2$. Por tanto, para calcular la intersección sustituimos x por y en la ecuación de la curva:

$$(3x - 1)^2 = \frac{x(x + 1)(2x + 1)}{6}.$$

Después de un poco de manipulación obtenemos una ecuación del tipo

$$x^3 - \frac{51}{2}x^2 + \cdots = 0$$

con lo cual la primera coordenada x del tercer punto satisface

$$1 + \frac{1}{2} + x = \frac{51}{2}.$$

Luego $x = 24$ e $y = 3 \cdot 24 - 2 = 70$. En otras palabras

$$1 + 2^2 + 3^2 + \cdots + 24^2 = 70^2.$$

Es decir, si el cuadrado tiene $70^2 = 4900$ pelotas podremos construir una pirámide con 24 pisos. Podríamos continuar encontrando muchas soluciones racionales de la ecuación $y^2 = \frac{x(x+1)(2x+1)}{6}$. Sin embargo, las únicas que tiene coeficientes naturales son $(1, 1)$ y $(24, 70)$.

6.6 El grupo de una curva elíptica

Teorema 6.16 *Sea F una cúbica no singular y $O \in V(F)$. Entonces la siguiente regla*

$$P + Q = O(PQ)$$

define una estructura de grupo abeliano en $V(F)$, en la que O es el neutro.

Demostración. *Parte trivial.* Como $PQ = QP$, la operación definida en el Teorema es conmutativa. Por otro lado, del Problema 6.5.9 tenemos $O + P = O(OP) = P$ y $P + (OO)P = O(P((OO)P)) = O(OO) = O$. Lo que demuestra que O es neutro de esta operación y $(OO)P$ es inverso de P.

Parte difícil. Por tanto, solo falta demostrar que la operación es asociativa, lo que se reduce a demostrar la siguiente igualdad

$$(PP')(QQ') = (PQ)(P'Q'), \quad (P, P', Q, Q' \in V(F)). \tag{6.5}$$

En efecto, supongamos que (6.5) se verifica para cada cuatro puntos P, P', Q y Q' de $V(F)$. Si $P, Q, R \in V(F)$ entonces aplicando el Problema 6.5.9 y (6.5) para los puntos PQ, O, Q y QR tenemos

$$P(Q+R) = P(O(QR)) = ((PQ)Q)(O(QR)) = ((PQ)O)(Q(QR)) = ((PQ)O)R = (P+Q)R$$

y, por tanto,

$$P + (Q+R) = O(P(Q+R)) = O((P+Q)R) = (P+Q) + R.$$

Para demostrar (6.5) podemos suponer sin pérdida de generalidad que K es algebraicamente cerrado. Eso implica que K es infinito, que es en realidad lo único que vamos a utilizar.

Consideramos la siguiente matriz 3×3, en la que falta una entrada por completar:

$$\begin{pmatrix} P & P' & PP' \\ Q & Q' & QQ' \\ PQ & P'Q' & \end{pmatrix}.$$

Obsérvese que los tres puntos de las filas y columnas que están completas son los tres puntos de intersección de F con una recta L, de forma que el número de veces que aparece un punto L en una fila (o columna) es $I(P, L \cap F)$. Si, siguiendo esa regla, completamos la entrada que falta, considerando que forma parte de la tercera fila, el punto que deberíamos colocar es $(PQ)(P'Q')$. Si, por el contrario, completamos la matriz atendiendo a que las columnas cumplan la propiedad mencionada deberíamos poner $(PP')(QQ')$ en la entrada vacía. Lo que hay que demostrar es que en ambos casos el resultado es el mismo.

Vamos a considerar dos casos que llamaremos degenerado y no degenerado respectivamente. En el caso degenerado se verificará que un punto aparece repetido en dos filas y columnas distintas, en una de las ocho posiciones rellenas de la matriz. Cuando esto no ocurra, es decir, en posiciones de diferentes filas y columnas no se repita ningún punto, diremos que estamos en el caso no degenerado. Empezaremos con el caso degenerado que es el más sencillo.

Caso degenerado. Por simetría, basta considerar tres subcasos: (1) $P = Q'$ (si el punto repetido aparece en filas y columnas completas), (2) $P = QQ'$ (si el punto repetido aparece en una filas completas y en una columna no completa) y (3) $PP' = PQ$ (si el punto repetido aparece en una fila no completa y en una columna no completa).

(1) Si $P = Q'$ entones $(PP')(QQ') = (PP')(QP) = (PQ)(P'P) = (PQ)(P'Q')$, por el Problema 6.5.9.

(3) Si $PP' = PQ$ entonces $P' = Q$, con lo que, cambiando los papeles de P y P' y Q y Q', podemos suponer que estamos en el caso (1).

(2) Si $P = QQ'$ entonces $PQ = Q'$ y por tanto, $(PP')(QQ') = (PP')P = P'$ y $(PQ)(P'Q') = Q'(P'Q') = P'$.

Caso no degenerado. Vamos a denotar por L'_i a la recta que contiene a los puntos de la fila i y L''_i a la recta que contiene a los puntos de la columna i. Denotaremos también por P_{ij}

al punto en la posición (i,j) de la matriz, siempre que $(i,j) \neq (3,3)$. Se entiende que si una fila o columna tiene un punto repetido entonces la correspondiente recta es la tangente a F en dicho punto. Definimos

$$\alpha(L'_i) = \sum I(X, L'_i, F), \quad \alpha''_i = \sum I(Y, L''_i, F)$$

donde X recorre los puntos que aparecen en la fila i e Y recorre los puntos de la columna i. Claramente, si $i \leq 2$ entonces $\alpha(L'_i) = \alpha(L''_i) = 3$.

Afirmación 1: $L'_i \neq L''_j$ para todo i,j. Si consideramos las posiciones de la matriz que están en la fila i o la columna j pero no en ambas, tenemos cuatro posiciones de las que al menos están rellenas tres. Si $L = L'_i = L''_j$, entonces los puntos de estas tres o cuatro posiciones completas están en $V(L) \cap V(F)$. Sea X el multiconjunto de los puntos que aparecen en la fila i (entendiendo que si el mismo punto aparece en dos posiciones diferentes, lo consideramos como dos puntos) pero no en la columna j e Y el conjunto de los puntos que aparecen en la columna j pero no en la columna i. Claramente $X \neq \emptyset \neq Y$ y $|X| \geq 2$ ó $|Y| \geq 2$. Además, por hipótesis $X \cap Y = \emptyset$. Luego $\sum_{R\in X} I(R,L,F) \geq 1 \leq \sum_{R\in Y} I(R,L,F)$, pero ambos no pueden ser iguales a 1, y $\sum_{R\in X} I(R,L,F) + \sum_{R\in Y} I(R,L,F) = \sum_{R\in X\cup Y} I(R,L,F) \leq 3$. (En las expresiones anteriores $R \in X$ no considera R repetido si aparece más de una vez en X sino una sola). Luego $\sum_{R\in X} I(P, L\cap F) = 1$ ó $\sum_{R\in Y} I(P, L\cap F) = 1$. Por simetría, podemos suponer que se verifica lo segundo. Eso implica que $i \neq 3 = j$. Llamemos k al elemento de $\{1,2\} \setminus \{i\}$. Entonces $P_{k3} \in V(L) \setminus \{P_{i1}, P_{i2}\}$. Eso implica que $P_{k3} = P_{i3}$ y, por tanto, $I(P_{i3}, L\cap F) \geq 2$. Concluímos que $P_{k3} = P_{i3} = P_{ih}$ para algún $h = 1$ ó 2, en contra de la hipótesis de no degeneración. Esto acaba la demostración de la Afirmación 1.

Consideraremos las cúbicas $L' = L'_1L'_2L'_3$ y $L'' = L''_1L''_2L''_3$. Como K es infinito, $L'_1(K)$ contiene un punto R' diferente de P, P' y PP' y un punto R'' que no está en $L'(K)$. Fijando una coordenadas homogéneas para R' y R'' podemos considerar el siguiente sistema lineal de ecuaciones

$$\begin{pmatrix} F(R') \neq 0 & L'(R') = 0 & L''(R') \\ F(R'') & L'(R'') \neq 0 & L''(R') \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

La matriz de los coeficientes tiene rango dos, con lo que la solución de este sistema define un único punto $[a, a', a'']$ de $\mathbb{P}^2(K)$. Definimos

$$F_0 = aF + a'L' + a''L''.$$

Esto implica que $F_0(R') = F_0(R'') = 0$.

Afirmación 2: $F_0 = 0$. Por reducción al absurdo suponemos que $F_0 \neq 0$.

Sea $L = L'_i$ y supongamos que un punto R aparece repetido N veces en la fila i de la matriz. Entonces $I(R,L,F) \geq N$, $I(R,L,L') = \infty$ y $I(R,L,L'') \geq N$. Entonces $I(R,L,F_0) \geq N$, por el Problema 6.4.6. El mismo argumento muestra que si $L = L''_i$ y R aparece repetido N veces en la columna i de la matriz entonces $I(P, L\cap F_0) \geq N$. Si $X = \{P, P', PP'\}$, entonces $\sum_{R\in X\cup\{R'\}} I(R, L'_1, F_0) = \sum_{R\in X} I(R, L'_1, F_0) + I(R', L'_1, F_0) \geq 3 + 1$. Del Corolario 6.11 deducimos que L'_1 divide a F_0. Pongamos $F_0 = L'_1C$.

Elijamos un punto R que aparece $N \geq 1$ veces en la segunda fila de la matriz. Si R no aparece en la primera fila entonces $I(R, L_2', L_1') = 0$ y por tanto $I(R, L_2', C) = I(R, L_2', F_0) \geq N$. De hecho esto también se verifica, aunque R aparezca en la primera fila. En efecto, si R aparece en la primera fila, entonces $N = 1$ por la hipótesis de no degeneración y R aparece dos veces en una columna (una en la primera fila y otra en la segunda). Si se trata de la columna j tendremos $I(R, L_j'', F_0) \geq 2$. Por la Afirmación 1, $I(R, L_j'', L_i') = 1$ y eso implica que $I(R, L_j'', C) \geq 1 = N$.

Del párrafo anterior se deduce que $\sum_{R \in \mathbb{P}^2(K)} I(R, L_2', C) \geq \sum_{R \in \mathbb{P}^2(K)} I(R, L_2', F) = 3$. Aplicando una vez más el Corolario 6.11 deducimos que L_2' divide a C. Por tanto, $F_0 = L_1' L_2' M$, con M una recta.

Vamos a demostrar que $M = L_3'$. Para ello consideraremos por separado los casos en que los dos puntos de la última columna son iguales o diferentes. Si $R = PQ = P'Q'$, entonces, de la hipótesis de no degeneración deducimos que R no aparece ni en la primera ni en la segunda fila. Luego $I(R, L_1' \cap L_3') = I(R, L_2' \cap L_3') = 0$. Pero sabemos que $I(P, L_3' \cap F_0) \geq 2$, con lo que $I(P, L_3' \cap M) \geq 2$. Eso implica que $L_3' = M$, como queríamos demostrar. Si por el contrario $PQ \neq P'Q'$, llamamos R a uno de ellos dos y sea N el número de veces que aparece en la misma columna, que supondremos que es la j. Entonces $I(R, L_j'', F_0) \geq N$ e $I(R, L_j'', L_1' L_2') = N - 1$ (pues $L_1' \neq L_j'' \neq L_2'$). Entonces $I(R, L_j'', M) = I(R, L_j'', F_0) - I(R, L_j'', L_1' L_2') \geq 1$, o sea $R \in M(K)$. Esto muestra que PQ y $P'Q'$ son dos puntos distintos de L_3' y de M, lo que muestra que $M = L_3'$ también en este caso.

En resumen $F_0 = cL'$, para algún $c \in K^*$. Como $F_0(R'') = 0 \neq L'(R'')$, obtenemos una contradicción que termina la demostración de la Afirmación 2.

Luego $aF + a'L' + a''L'' = 0$. Si $a = 0$ entonces $a'L_1' L_2' L_3' = a'L' = -a''L'' = -a''L_1'' L_2'' L_3''$, con $a'a'' \neq 0$ y eso contradiría la Afirmación 1. Por tanto, $F = c'L' + c''L''$ con $c', c'' \in K$. Como F es irreducible, por ser no singular, necesariamente $c'c'' \neq 0$. Eso implica que $V(F) \cap L'(K) = V(F) \cap L''(K)$.

Vamos a denotar con T al punto de $L_3'(K) \cap L_3''(K)$. Por la Afirmación 1, este punto es único. Entonces $F(T) = 0$, con lo que

$$T \in V(F) \cap L_3'(K) \cap L_3''(K) \subseteq \{PP', QQ', (PP')(QQ')\} \cap \{PQ, P'Q', (PQ)(P'Q')\}.$$

Por la hipótesis de no degeneración, los dos primeros puntos de cada uno de los dos primeros conjuntos son diferentes a los dos primeros puntos del otro.

Supongamos que no se verifica (6.5). Entonces se verifica uno de las siguientes casos: (1) $T = PQ = (PP')(QQ')$, (2) $T = P'Q' = (PP')(QQ')$, (3) $T = PP' = (PQ)(P'Q')$ ó (4) $T = QQ' = (PQ)(P'Q')$. Por simetría basta considerar el primer caso. Supongamos pues que $T = PQ = (PP')(QQ')$ y sea $P_1 = (PQ)(P'Q') \in V(F) \cap L_3'(K) \subseteq V(F) \cap L'(K) = V(F) \cap L''(K)$. Como por hipótesis $P_1 \neq T$, necesariamente $P_1 \notin L_3''(K)$. Si $P_1 \in L_1''(K)$ entonces $P_1 \in L_1''(K) \cap L_3'(K) = \{PQ\}$. Eso implica que $(PQ)(P'Q') = P_1 = PQ = (PP')(QQ')$, en contra de que estamos suponiendo que (6.5) no se verifica. Por tanto, $P_1 \in L_2''(K) \cap L_3'(K) = \{P'Q'\}$, es decir $(PQ)(P'Q') = P'Q'$. Esto implica que $I(P'Q', L_3' \cap F) \geq 2$. Por otro lado, $PQ = T \in L_3'(K) \cap L_3''(K) \cap L_1''(K)$. Por tanto $I(PQ, L_3' \cap L'') \geq 2$. Como además $I(PQ, L_3' \cap L') = \infty$, deducimos que $I(PQ, L_3' \cap F) \geq 2$. Como $PQ = (PP')(QQ') \neq (PQ)(P'Q') = P'Q'$, concluimos que $\sum_{R \in \mathbb{P}^2(K)} I(R, L_3', F) \geq I(PQ, L_3' \cap F) + I(P'Q', L_3' \cap F) \geq 4$, lo que proporciona una contradicción. ∎

Sea F la curva de Weierstrass dada por la Ecuación de Weierstrass

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

y fijemos como punto $O = [0, 1, 0]$ para definir la suma $P + Q = O(PQ)$. Salvo que se diga lo contrario, cuando mencionemos el grupo de una curva de Weierstrass nos estaremos refiriendo a este caso.

Vamos ahora a ver cómo obtener fórmulas precisas para la suma en función de las coordenadas. Sabemos que O es el único punto de la curva en la recta del infinito, por tanto nos referiremos a O como punto del infinito de la curva. Los demás puntos de la curva los representaremos por sus coordenadas afines. Es decir, $[x, y, 1]$ se representa como (x, y). Sabemos que

$$(x, y) + O = (x, y).$$

Como O es un punto de inflexión de F, tenemos $OO = O$. En la demostración del Teorema 6.16 hemos visto que $-P = (OO)P$, luego

$$-P = OP.$$

La recta que pasa por $P = (x_1, y_1) = [x_1, y_1, 1]$ y $O = [0, 1, 0]$ es $X - x_1Z$. Por tanto, $-P = (x_1, y_2)$ con y_1 e y_2 las soluciones de la ecuación

$$Y^2 + (a_2x_1 + a_3)Y = x_1^3 + a_2x_1^2 + a_4x_1 + a_6$$

Por tanto $y_1 + y_2 = -(a_2x_1 + a_3)$. En resumen

$$-(x_1, y_1) = (x_1, -(y_1 + a_2x_1 + a_3)). \tag{6.6}$$

En particular,

$$(x_1, y_1) + (x_2, y_2) = O \quad \Leftrightarrow \quad \begin{cases} x_1 = x_2 \\ \text{ e} \\ y_1 + y_2 + a_2x_1 + a_3 = 0. \end{cases} \tag{6.7}$$

Falta obtener fórmulas para $(x_1, y_1) + (x_2, y_2)$ para el caso en que las condiciones de (6.7) no se verifiquen. Pongamos $P_i = (x_i, y_i)$. Podemos mirar el caso en que $P_1 + P_2 = O$ con una perspectiva geométrica. En este caso, la recta que pasa por P_1, P_2 y O es $X - x_1Z$, es decir, la recta afín es una recta vertical: $X = x_1$. En los demás casos, la recta afín tiene una ecuación del tipo $Y = mX + n$ y, de hecho, como pasa por P_1 tiene una ecuación $Y - y_1 = m(X - y_1)$. Determinar el valor de m es fácil:

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{si } P_1 \neq P_2; \\ \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_2y_1}{2y_1 + a_2x_1 + a_3}, & \text{si } P_1 = P_2. \end{cases} \tag{6.8}$$

Dejamos al lector que verifique esto con la indicación de que si $P_1 = P_2$, entonces m es la derivada de y respecto de x, que se puede obtener derivando de forma implícita en la Ecuación

de Weierstrass. Obsérvese que estas fórmulas tienen utilidad incluso en el caso en que el denominador de la expresión sea 0. Por ejemplo, cuando $P_1 \neq P_2$ y el denominador $x_2 - x_1$ es cero, entonces P_1, P_2 y O están en la recta $X - x_1 Z$ y tenemos $P_1 P_2 = O$ y por tanto $P_1 + P_2 = OO = O$. Si $P_1 = P_2$, que el denominador $2y_2 + a_2 x_2 + a_3$ se anule significa que se verifican las condiciones de (6.7) y de nuevo tenemos $P_1 + P_2 = O$. Por tanto,

$$m \in K \quad \Leftrightarrow \quad P_1 + P_2 \neq O. \tag{6.9}$$

En caso contrario, para calcular $P_1 P_2$ sustituimos $Y = y_1 + m(X - x_1) = mX + n$ en la ecuación de Weierstrass lo que nos proporcionará una ecuación de tercer grado en X:

$$X^3 - s_1 X^2 + \cdots = 0.$$

Una sencilla cuenta muestra que $s_1 = m^2 + a_1 m - a_2$. Por tanto, si $P_1 P_2 = (x_3, z)$, entonces

$$x_1 + x_2 + x_3 = s_1 = m^2 + a_1 m - a_2,$$

y si $z = y_1 + m(x_3 - x_1)$ y $P_1 + P_2 = O(P_1 P_2) = -(P_1 P_2) = -(x_3, z)$. Utilizando (6.6) obtenemos que la segunda coordenada de $P_1 + P_2$ es $y_3 = -(z + a_2 x_3 + a_3) = -(y_1 + m(x_3 - x_1) + a_2 x_3 + a_3)$. En resumen

Proposición 6.17 *Sean $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ dos puntos afines de la curva elíptica dada por la Ecuación de Weierstrass afín $y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$. Para obtener $P_1 + P_2$ primero calculamos m de acuerdo con la fórmulas de (6.8), entendiendo que si el denominador de la expresión es cero entonces $m = \infty$. Si $m = \infty$ entonces $P_1 + P_2 = O$. En caso contrario $P_1 + P_2 = (x_3, y_3)$, con*

$$\begin{aligned} x_3 &= m^2 + a_1 m - a_2 - x_1 - x_2, \\ y_3 &= -(y_1 + m(x_3 - x_1) + a_2 x_3 + a_3). \end{aligned}$$

6.7 ¿Qué ecuaciones de Weierstrass definen curvas elípticas?

Sigue pendiente el problema de decidir qué ecuaciones de Weierstrass definen curvas elípticas. Para atacar este problema vamos a definir otras dos cúbicas afines que están fuertemente relacionadas con la ecuación (6.4). Se trata de

$$y^2 = 4x^3 + b_2 x^2 + 2b_4 x + b_6 \tag{6.10}$$

y

$$y^2 = x^3 - 3^3 c_4 x - 2 \cdot 3^3 c_6. \tag{6.11}$$

donde

$$\begin{aligned} b_2 &= a_1^2 + 4a_2 \\ b_4 &= 2a_4 + a_1 a_3 \\ b_6 &= a_3^2 + 4a_6 \\ c_4 &= b_2^2 - 2^3 3 a_6 \\ c_6 &= -b_2^3 + 2^2 3^3 b_2 b_4 - 2^3 3^3 b_6. \end{aligned}$$

También introducimos otros dos números que van a representar un papel importante:

$$\begin{aligned} b_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2, \\ \Delta &= -b_2^2b_8 - 2^3b_4^3 - 3^3b_6^2 + 3^2b_2b_4b_6. \end{aligned} \tag{6.12}$$

Δ se llama *discriminante* de la curva de Weierstrass (6.3). Antes de nada, observamos que se verifican las siguientes igualdades:

$$4b_8 = b_2b_6 - b_4^2 \tag{6.13}$$

$$2^6 3^3 \Delta = c_4^3 - c_6^2. \tag{6.14}$$

Veamos ahora para qué nos sirve toda esta notación. En primer lugar, si K tiene característica diferente de 2 podemos completar cuadrados en la parte de la izquierda de la ecuación (6.4). Eso equivale a realizar el cambio

$$y \mapsto \frac{y - (a_1x + a_3)}{2} \tag{6.15}$$

que transforma la ecuación (6.4) en la ecuación (6.10). Esto implica que en característica 2 toda curva elíptica es equivalente a una Curva de Weierstrass cuya curva afín asociada viene dada por una ecuación del tipo $y^2 = P(X)$, donde $P(X)$ es un polinomio de tercer grado.

Si además K tiene característica diferente de 3 podemos completar cubos en la parte izquierda de (6.10). Esto equivale a realizar los siguientes cambios:

$$x \rightarrow \frac{x - 3b_2}{2^2 \cdot 3^3} \tag{6.16}$$

$$y \rightarrow \frac{y}{2^2 \cdot 3^3} \tag{6.17}$$

y lo que obtenemos es precisamente la ecuación (6.11). Por tanto toda curva elíptica definida sobre un cuerpo de característica diferente de 2 y 3 viene dada por una *Ecuación de Weierstrass reducida*:

$$Y^3Z = X^3 + AXZ^2 + BZ^3.$$

Pongamos lo aprendido en la siguiente proposición.

Proposición 6.18 *Sea E una curva de Weierstrass sobre K.*

(1) Si K tiene característica diferente de 2 entonces E es equivalente sobre K a una curva cuya ecuación afín tiene la forma $Y^2 = P(X)$, para P un polinomio en $K[X]$ de grado 3.

(2) Si además K tiene característica diferente de 3 entonces se puede elegir P de la forma $X^3 + AX + B$, con $A, B \in K$.

Podemos ahora dar respuesta a la pregunta que da título a esta sección.

Teorema 6.19 *Una curva de Weierstrass define una curva elíptica si y solo si su discriminante es diferente de 0.*

Para demostrar el Teorema 6.19 necesitamos cierta preparación. Comenzamos con el siguiente lema.

Lema 6.20 *Si K tiene característica diferente de 2 y $f \in K[X]$ es un polinomio de grado 3 entonces la curva proyectiva con curva afín asociada $y^2 = f(X)$ es no singular si y solo si f no tiene raíces múltiples.*

Demostración. Si $f = aX^3 + bX^2 + cX + d$, entonces la curva proyectiva es

$$F = Y^2Z - aX^3 - bX^2Z - cXZ^2 - dZ^3 = Y^2Z - X^3 f\left(\frac{Z}{X}\right) = Y^2Z - Z^3 f\left(\frac{X}{Z}\right).$$

Los puntos singulares de F son las soluciones del siguiente sistema de ecuaciones:

$$\begin{aligned} Y^2Z &= aX^3 + bX^2Z + cXZ^2 + dZ^3, \\ Z^2 f'\left(\frac{X}{Z}\right) &= 0, \\ 2YZ &= 0, \\ Y &= X^2 f'\left(\frac{Z}{X}\right). \end{aligned}$$

Si $Z = 0$ entonces $X = 0$ (por la primera ecuación) y eso implica que $Y = 0$, por la última ecuación. Por tanto, la recta del infinito no tiene puntos singulares. Si $Z \neq 0$, podemos suponer que $Z = 1$. Como K tiene característica diferente de 2, necesariamente $Y = 0$ y las ecuaciones que obtenemos son

$$f(x) = f'(x) = 0, \quad x^2 f'\left(\frac{1}{x}\right) = 0.$$

Las dos primeras igualdades son equivalentes a que x sea una raíz múltiple de f. La última es consecuencia de las otras dos pues

$$3f(x) - xf'(x) = 3(ax^3 + bx^2 + cx + d) - x(3ax^2 + 2bx + c) = bx^2 + 2cx + d = x^2 f'\left(\frac{1}{x}\right).$$

∎

Recordemos que el discriminante de un polinomio $f(X) \in K[X]$ de grado n cuyas raíces en la clausura algebraica son $\alpha_1, \ldots, \alpha_n$ (contando repetidas las raíces múltiples según sus multiplicidades) es

$$D(f) = \prod_{i<j} (\alpha_i - \alpha_j)^2 = \det(V(\alpha_1, \ldots, \alpha_n))^2 = \det(V(\alpha_1, \ldots, \alpha_n) V(\alpha_1, \ldots, \alpha_n)^T) \qquad (6.18)$$

donde

$$V(\alpha_1, \ldots, \alpha_n) = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \ldots & \alpha_n^{n-1} \end{pmatrix},$$

la matriz de Vandermonde. Si ponemos

$$\sigma_k = \sum_{i=1}^{k} \alpha_i^k,$$

entonces la entrada (i,j) de $V(\alpha_1,\ldots,\alpha_n)V(\alpha_1,\ldots,\alpha_n)^T$ es

$$\sum_{k=1}^{n} \alpha_k^{i-1}\alpha_k^{j-1} = \sigma_{i+j-2}.$$

O sea

$$V(\alpha_1,\ldots,\alpha_n)V(\alpha_1,\ldots,\alpha_n)^T = \begin{pmatrix} \sigma_0 & \sigma_1 & \sigma_2 & \ldots & \sigma_{n-1} \\ \sigma_1 & \sigma_2 & \sigma_3 & \ldots & \sigma_n \\ \sigma_2 & \sigma_3 & \sigma_4 & \ldots & \sigma_{n+1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \sigma_{n-1} & \sigma_n & \sigma_{n+1} & \ldots & \sigma_{2n-2} \end{pmatrix} \tag{6.19}$$

Por ejemplo, para un polinomio $f(X) = X^3 - s_1X^2 + s_2X - s_3$ tenemos

$$\begin{aligned} \sigma_1 &= s_1 \\ \sigma_2 &= s_1^2 - 2s_2 \\ \sigma_3 &= s_1^3 - 3s_1s_2 + 3s_1 \\ \sigma_4 &= s_1^4 - 4s_1^2s_2 + 2s_2^2 + 4s_1s_3 \end{aligned} \tag{6.20}$$

De (6.18), (6.19) y (6.22) deducimos que

$$D(f) = \begin{vmatrix} \sigma_0 & \sigma_1 & \sigma_2 \\ \sigma_1 & \sigma_2 & \sigma_3 \\ \sigma_2 & \sigma_3 & \sigma_4 \end{vmatrix} = \begin{vmatrix} 3 & s_1 & s_1^2 - 2s_2 \\ s_1 & s_1^2 - 2s_2 & s_1^2 - 2s_2 \\ s_1^2 - 2s_2 & s_1^2 - 2s_2 & s_1^4 - 4s_1^2s_2 + 2s_2^2 + 4s_1s_3 \end{vmatrix}.$$

Desarrollando este determinante obtendremos una expresión de $D(P)$ en términos de s_1, s_2 y s_3. En general esta expresión es bastante complicada como para recordarla (Problema 18). El caso particular en que el polinomio tiene la forma $X^3 + pX + q$ es sencillo y útil. En tal caso

$$\begin{aligned} \sigma_1 &= 0 \\ \sigma_2 &= -2p \\ \sigma_3 &= -3q \\ \sigma_4 &= 2p^2 \end{aligned}$$

y por tanto el discriminante de este polinomio es

$$\begin{pmatrix} 3 & 0 & -2p \\ 0 & -2p & -3q \\ -2p & -3q & 2p^2 \end{pmatrix} = -4p^3 - 27q^2.$$

Por tanto

$$D(X^3 + pX + q) = -4p^3 - 27q^2 \tag{6.21}$$

Con esta información estamos en condiciones de demostrar el Teorema 6.19.

Demostración del Teorema 6.19. Consideramos primero el caso en que K tiene característica diferente de 2 y 3. Como las ecuaciones (6.4), (6.10) y (6.11) definen curvas equivalentes sobre $\overline{K}$, si una de ellas define una curva no singular entonces las otras también. Por el Lema 6.20, (6.11) es no singular si y solo si el discriminante del polinomio $X^3 - 3^3c_4X - 2 \cdot 3^3c_6$ es diferente de 0. Calculando este discriminante con (6.21) obtenemos que (6.3) define una curva elíptica si y solo si $2^2 \cdot 3^9(c_4^3 - c_6^2) = 2^8 \cdot 3^{12}\Delta \neq 0$ si y solo si $\Delta \neq 0$.

Si K tiene característica 3 entonces no podemos utilizar la ecuación (6.11), pero sí la la ecuación (6.10), que en característica 3 toma la forma

$$Y^2 = X^3 + b_2X^2 - b_4X + b_6.$$

En este caso, teniendo en cuenta que K tiene característica 3, de las fórmulas de (6.22) obtenemos los siguientes valores de los σ_i para el polinomio $P(X) = X^3 + b_2X^2 - b_4X + b_6$:

$$\begin{aligned}
\sigma_1 &= -b_2 \\
\sigma_2 &= b_2^2 - b_4 \\
\sigma_3 &= -b_2^3 \\
\sigma_4 &= b_2^4 + b_2^2b_4 - b_4^2 + b_2b_6
\end{aligned}$$

y por tanto,

$$\begin{aligned}
D(X^3 + b_2X^2 - b_4X + b_6) &= \begin{vmatrix} 0 & \sigma_1 & \sigma_2 \\ \sigma_1 & \sigma_2 & \sigma_3 \\ \sigma_2 & \sigma_3 & \sigma_4 \end{vmatrix} = -\sigma_1\sigma_2\sigma_3 - \sigma_1^2\sigma_4 - \sigma_2^3 \\
&= -b_2^4(b_2^2 - b_4) - b_2^2(b_2^4 + b_2^2b_4 - b_4^2 + b_2b_6) - (b_2^2 - b_4)^3 \\
&= b_2^2(b_4^2 - b_2b_6) + b_4^3 = -b_2^2b_8 + b_4^3 = \Delta.
\end{aligned}$$

En la última línea se ha utilizado (6.13). Del Lema 6.20, deducimos de nuevo que (6.10) es no singular si y solo si $\Delta \neq 0$.

Si K tiene característica 2 no podemos utilizar ni la ecuación (6.10) ni la (6.11). Pero en este caso, la expresión de Δ se simplifica algo:

$$\Delta = b_2^2b_8 + b_6^2 + b_2b_4b_6 = a_1^6a_6 + a_1^5a_3a_4 + a_1^4a_2a_3^2 + a_1^4a_4^2 + a_3^4 + a_1^2a_3^2 \tag{6.22}$$

Del Problema 6.7.19 tenemos que el único punto de la recta del infinito que está en la curva es no singular. Por tanto solo tenemos que analizar la singularidad en los puntos de la forma $[x, y, 1]$. Tomando derivadas en la curva de Weierstrass $F = Y^2Z + a_1XYZ + a_3YZ^2 + X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$ obtenemos que los puntos singulares de la forma $[x, y, 1]$ son los que satisfacen las siguientes igualdades:

$$\begin{aligned}
y^2 + a_1xy + a_3y + x^3 + a_2x^2 + a_4x + a_6 &= 0 \\
a_1y + x^2 + a_4 &= 0 \\
a_1x + a_3 &= 0 \\
y^2 + a_1xy + a_2x^2 + a_6 &= 0
\end{aligned}$$

Se observa fácilmente que la última ecuación es redundante. Si $a_1 = 0$ el sistema tendrá una solución en $\overline{K}$ si y solo si $a_3 = 0$. En este caso los puntos singular se obtienen resolviendo el sistema

$$\begin{aligned} y^2 &= x^3 + a_2x^2 + a_4x + a_6 \\ x^2 &= a_4 \end{aligned}$$

que tiene solución en $\overline{K}$. Por tanto, si $a_1 = 0$ la curva es singular y de (6.22) tenemos que $\Delta = 0$. Si $a_1 \neq 0$ entonces el único punto singular posible de la forma $[x, y, 1]$ es

$$\begin{aligned} x &= a_3a_1^{-1} \\ y &= a_3^2a_1^{-3} + a_1^{-1}a_4 \end{aligned}$$

Sustituyendo en la ecuación de Weierstrass, después de algunas cuentas en las que utilizaremos (6.22), deducimos que dicho punto satisface la Ecuación de Weierstrass si y solo si

$$0 = a_1^{-6}\Delta.$$

Esto acaba la demostración. ∎

6.8 Factorización con curvas elípticas

Vamos ahora a ver un método para factorizar enteros utilizando curvas elípticas descubierto por a H.W. Lenstra en 1986. Recordemos que el método $p-1$ de Pollard se basaba en que un divisor primo p de n tuviera la propiedad de que $p-1$ no tuviera divisores primos grandes. El método de Lenstra es una variación de este método pero en lugar de depender de los ordenes de los grupos $\mathbb{Z}_p^*$, para p un divisor primo de n, que solo tenemos una cantidad pequeña, vamos a depender de las curvas elípticas sobre $\mathbb{F}_p$, que es un universo mucho más grande.

Recordemos que dado un número primo p y un número entero no nulo a denotamos por $v_p(a)$ a la multiplicidad de p como divisor de a, es decir $v_p(a)$ viene determinado por

$$a = p^{v_p(a)}b, \text{ con } \operatorname{mcd}(p, b) = 1.$$

Extenderemos esta aplicación v_p para que esté definida en $\mathbb{Q}$ poniendo

$$v_p\left(\frac{a}{b}\right) = v_p(a) - v_p(b) \quad \text{y} \quad v_p(0) = \infty.$$

El conjunto

$$A_p = \{a \in \mathbb{Q} : v_p(a) \geq 0\}$$

es un subanillo de $\mathbb{Q}$ formado por las fracciones cuyo denominador es coprimo con p y sus ideales no nulos forman una cadena estrictamente decreciente

$$A_p \supset pA_p \supset p^2A_p \supset \dots$$

Por tanto, los elementos invertibles de A_p son las fracciones en las que el numerador y el denominador son coprimos con p. Este anillo se llama *localizado* de $\mathbb{Z}$ en p.

Si n es un entero arbitrario, ponemos

$$A_n = \bigcap_{p|n, p\in\mathbb{P}} A_p,$$

donde $\mathbb{P}$ denota el conjunto de los números primos. Claramente A_n está formado por las fracciones cuyo denominador es coprimo con n y sus elementos invertibles son las fracciones en las que el numerador también es coprimo con n.

Para simplificar, pongamos $A = A_n$. Dados $x_1, x_2 \in A$, ponemos $x_1 \equiv x_2 \mod n$ si $x_1 - x_2 \in A_n$, es decir la relación es la relación de congruencia definida por el ideal An generado por n en A. Obsérvese que $x \equiv y \mod n$ precisamente si el numerador de una fracción irreducible que define $x - y$ es múltiplo de n. Por tanto la relación efectivamente se restringe a la relación de congruencia módulo n en $\mathbb{Z}$. Más aún, si $a, b, c \in \mathbb{Z}$ con $\mcd(b, n) = 1$, entonces $\frac{a}{b} \equiv c \mod n$ precisamente si $a \equiv bc \mod n$. Esto implica que $A/An \simeq \mathbb{Z}_n$, es decir, cada clase de equivalencia viene determinada por un único número natural menor que n. Denotaremos por $x \mod n$ al representante canónico en $\mathbb{Z}_n$ de un elemento $x \in A$.

Consideremos una curva elíptica racional E dada por una ecuación de Weierstrass simplificada $Y^2 = X^3 + aX + b$ con coeficientes a y b enteros y un punto $P \in \mathbb{Z}^2$ de esta curva. Una forma de elegir una curva así es seleccionar tres enteros $x, y, a \in \mathbb{Z}$ al azar y poner $P = (x, y)$ y $b = y^2 - x^3 - ax$. Sólo tenemos que asegurarnos de que la curva es no singular, lo que equivale a que el polinomio $X^3 + aX + b$ no tenga raíces múltiples y esto a su vez equivale a que

$$4a^3 + 27b^2 \neq 0.$$

En realidad nos interesa reducir la curva elíptica elegida módulo p y, por tanto exigiremos además

$$4a^3 + 27b^2 \not\equiv 0 \mod p$$

para que la reducción módulo p de la curva también sea no singular. Claro que esto no podemos comprobar porque no conocemos p. En la práctica calculamos $\mcd(4a^3 + 27b^2, n)$. Si este número es n, entonces desechamos esta curva, si es un divisor propio de n, hemos encontrado un divisor propio, que en realidad es nuestro objetivo. En el caso que queda, $\mcd(4a^3 + 27b^2, n) = 1$, se cumple la condición deseada.

Vamos a denotar por $+$ a la suma en E, considerada como curva racional. Además denotamos por E_p a la curva elíptica con coeficientes en $\mathbb{F}_p$ que se obtiene al reducir E módulo p y por $+_p$ a la suma en esta curva.

También consideraremos los puntos de la curva E que tienen coordenadas en A. Obsérvese que no podemos asegurar que sumar dos puntos de A^2 siga estando en A^2. ¿Cuándo nos salimos de A^2 al sumar dos puntos de A^2?

Proposición 6.21 *Sea E una curva elíptica con ecuación de Weierstrass $Y^2 = X^3 + aX + b$, con $a, b \in \mathbb{Z}$ y sea $n \in \mathbb{Z}$ tal que $\mcd(4a^3 + 27b^2, n) = 1$. Sean P_1 y P_2 dos puntos de E con coordenadas en A y $P_1 + P_2 \neq O$. Entonces las coordenadas de $P_1 + P_2$ están en A precisamente si para todo primo p que divide a n, $P_1 +_p P_2 \neq O$ en E_p.*

Demostración. Obsérvese que podemos suponer que n es una potencia de un primo p y que por tanto $A = A_p$. Pongamos $P_i = (x_i, y_i) \in A^2$ y $P_1 + P_2 = (x_3, y_3)$. Distinguiremos varios casos.

(a) Supongamos primero que $x_1 \not\equiv x_2 \mod p$ y por tanto $P_1 \neq P_2$ en E_p. Entonces $x_1 - x_2$ es invertible en A y en $A/Ap \simeq \mathbb{F}_p$. Luego las siguientes operaciones son posibles en A y en $\mathbb{F}_p$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \quad y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3).$$

Entonces $P_1 + P_2 = (x_3, y_3)$ y $P_1 +_p P_2$ es el resultado de reducir esto módulo p. Por tanto, en este caso $P_1 + P_2 \in A^2$ y $P_1 +_p P_2 \in \mathbb{F}_p^2$, o sea $P_1 +_p P_2 \neq O$.

(b) Supongamos ahora que $x_1 = x_2$. Como suponemos que $P_1 + P_2 \neq O$, entonces $y_1 + y_2 \neq 0$, con lo que $P_1 = P_2$ e $y_1 \neq 0$.

Supongamos que y_1 es múltiplo de p. Entonces la recta tangente a E_p en P_1 es $X = x_1$ y corta a E_p en P_1 como punto doble y en O como punto simple. Por tanto, $P_1 +_p P_2 = O$. Por otro lado como E_p es no singular, el polinomio $X^3 + aX + b$ no tiene puntos múltiples en $\mathbb{F}_p$ y por tanto $3x_1^2 + a$ no es múltiplo de p. Como $P_1 + P_2 = (x_3, y_3)$ con

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1, \quad y_3 = -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3),$$

se deduce que $P_1 + P_2 \notin A$.

Supongamos ahora que y_1 no es múltiplo de p. Entonces el cálculo anterior muestra que $P_1 + P_2 \in A$ y que $P_1 +_p P_2 \neq O$.

(c) Finalmente supongamos que $x_1 \neq x_2$ pero $x_1 \equiv x_2 \mod p$. Entonces

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \quad y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3).$$

Por tanto, $P_1 + P_2 \in A$ precisamente si $r = v_p(x_2 - x_1) \leq v_p(y_2 - y_1)$. Pongamos $x_2 - x_1 = p^r x$, con $x \in A$. Entonces, como $r \geq 1$, se tiene que $2r \geq r + 1$ y por tanto

$$\begin{aligned} y_2^2 &= x_2^3 + ax_2 + b \\ &= (x_1 + p^r x)^3 + a(x_1 + p^r x) + b \\ &= x_1^3 + ax_1 + b + p^r x(3x_1^2 + 3x_1 p^r x + p^{2r} x^2 + a) \\ &\equiv y_1^2 + p^r x(3x_1^2 + a) \mod p^{r+1} \end{aligned}$$

Eso implica que

$$v_p(y_2 - y_1) + v_p(y_1 + y_2) = v_p(y_2^2 - y_1^2) \geq r$$

y

$$v_p(y_2 - y_1) + v_p(y_1 + y_2) = r \Leftrightarrow v_p(3x_1^2 + a) = 0.$$

En particular $y_1^2 \equiv y_2^2 \mod p$. Por tanto, si $y_1 \equiv 0 \mod p$, entonces $y_2 \equiv 0 \mod p$ y $P_1 +_p P_2 = O$. Además en tal caso, $v_p(y_1 + y_2) \geq 1$ y $v_p(3x_1^2 + a) = 0$, pues $X^3 + aX + b$ no tiene raíces múltiples en un cuerpo de característica p. Luego $v_p(y_2 - y_1) < r$ y por tanto $P_1 + P_2 \notin A$. Supongamos que $y_1 \not\equiv 0 \mod p$, con lo que $y_2 \not\equiv 0 \mod p$. Si $y_1 + y_2 \equiv 0$

mod p, entonces $P_1 +_p P_2 = O$ e $y_1 \not\equiv y_2 \mod p$, por ser p impar. Luego $v_p(y_2 - y_1) = 0$ y por tanto $P_1 + P_2 \notin A$. Finalmente si $y_1 + y_2 \not\equiv 0 \mod p$, entonces $P_1 +_p P_2 \neq O$ y $v_p(y_2 - y_1) \geq r$, con lo que $P_1 + P_2 \in A$. ∎

El algoritmo de factorización de un número n lo que hace es calcular kP para $k = 1, 2, \ldots$ y $P = (x, y) \in \mathbb{Z}^2$ un punto de una curva elíptica con coeficientes enteros. Elegimos la curva elíptica cogiendo $a \in \mathbb{Z}$ al azar y $b = y^2 - x^3 - ax$. Si $\mathrm{mcd}(4a^3 + 27b^2, n) = n$, desechamos P ó a. Si $\mathrm{mcd}(4a^3 + 27b^2, n)$ es un divisor propio de n diferente de 1, entonces hemos encontrado un factor. En caso contrario, todas las curvas E_p, para p un divisor primo de n, son elípticas y calculamos kP hasta que tal punto se salga de A. En ese momento algún divisor de n se revelará, ya que en tal caso $kP = O$ en alguna curva E_p. Es decir si $P_1 = (k-1)P = (x_1, y_1) \neq O$, entonces $P +_p P_1 = O$. Eso quiere decir que $x_1 \equiv x \mod p$ e $y + y_1 \equiv 0 \mod p$.

En realidad tal vez encontramos antes el divisor, ya que al intentar calcular $P + (k-1)P$ en alguno de los pasos, tenemos que intentar calcular el inverso de $x_1 - x_2$ ó de $2y_1$. Mientras que este elemento sea coprimo con n, este inverso se encuentra dentro de A. En el momento en el que alguno de estos elementos no sea coprimo con n probablemente hemos encontrado un divisor propio de n.

Obsérvese la similitud de este método con el método $(p-1)$ de Pollard. En ambos casos nuestra esperanza en encontrar un elemento de orden bajo en un grupo. En aquel método el grupo era $\mathbb{Z}_p^*$ para alguno de los divisores primos de n. En éste es alguna de las curvas E_p. La ventaja de este segundo sobre el primero es que disponemos de mucho más grupos donde probar.

6.9 Tareas

(1) Sean F y G polinomios homogéneos. Demostrar que F divide a G si y solo si el F° divide a G°.

(2) Demostrar que si K es infinito y $f, g \in K[X, Y]$ entonces f y g tienen los mismos factores irreducibles en $K[X, Y]$ si y solo si definen la misma curva afín, es decir $\{(x, y) \in K^2 : f(x, y) = 0\} = \{(x, y) \in K^2 : g(x, y) = 0\}$. Demostrar un resultado similar para curvas proyectivas.

(3) Demostrar que si $F \in K[X, Y, Z]$ es un polinomio homogéneo de grado d y $P = (a, b, c)$ entonces

$$F'_X(P)a + F'_Y(P)b + F'_Z(P)c = dF(P).$$

Deducir que si P es un punto regular de $V(F)$ entonces P está en la recta tangente a F en P.

(4) Sean F y G dos curvas proyectivas y sea $P \in \mathbb{P}^2(K)$. Demostrar que P es un punto regular de FG si y solo si P es un punto regular de F y $G(P) \neq 0$, o viceversa.

(5) Demostrar que el orden de intersección no depende de la homografía utilizada para calcularla.

(6) Sean F y G polinomios homogéneos y L una recta. Demostrar que si L no divide a F ni a G entonces

$$I(P, L \cap (F+G)) \geq \min\{I(P, L \cap F), I(P, L \cap G)\}$$

y

$$I(P, L \cap F_1 F_2) = I(P, L \cap F_1) + I(P, L \cap F_2).$$

(7) Sea F una curva dada por una Ecuación de Weierstrass. Demostrar que el único punto de $V(F)$ en la recta del infinito es $P = [0, 1, 0]$ y que $I(P, Z \cap F) = 3$.

(8) Consideremos la curva de Weierstrass general (6.3) sobre un cuerpo de característica diferente de 2. Sean $P = (x_1, y_1)$ y $Q = (x_2, y_2)$ dos puntos afines de esta curva. Calcular fórmulas para el producto cuerda-tangente PQ en función de las coordenadas de P y Q. (Indicación para el caso en que $P = Q$: Si la recta tangente tiene la forma $y = mx + n$, entonces $m = y'$, la derivada de y respecto de x, que puede calcularse derivando de forma implícita en la Ecuación de Weierstrass.

(9) Demostrar las siguientes propiedades del producto secante tangente:

(a) $PQ = QP$.

(b) $P(PQ) = Q$.

(c) $PP = P$ sí y solo si P es un punto de inflexión.

(10) Utilizar el producto cuerda-tangente para obtener más puntos de la curva del Ejemplo 6.14.

(11) Sea F una curva de Weierstrass y $X = \{P \in V(F) : 2P = O\}$.

(a) Demostrar que X es un subgrupo de $V(F)$ y calcular X en función de la Ecuación de Weierstrass.

(b) Demostrar que si K tiene característica diferente de 2 entonces $X \cong \mathbb{Z}_2 \times \mathbb{Z}_2$.

(c) Demostrar que si K tiene característica 2 entonces $X = \{O\}$ ó $X \cong \mathbb{Z}_2$. (Indicación: Comparar con el Problema 17.)

(12) Completar los pasos de la obtención de las fórmulas para la suma en una curva de Weierstrass que no se han detallado.

(13) Existe una forma alternativa de demostrar la asociatividad de la suma definida en el Teorema 6.16. Simplemente obtener fórmulas precisas de la suma en función de las coordenadas y verificar la asociatividad aplicando estas fórmulas. El objetivo de este Problema es que el lector o lectora se convenza de que este camino no es más operativo ni siquiera en el caso más simple. Para ello considérese la curva elíptica dada por una Ecuación de Weierstrass reducida $Y^2 = X^3 + AX + B$ y fíjese $O = [0, 1, 0]$ para definir la suma. Comprobar la asociatividad en este caso utilizando la fórmula de la suma.

(14) La igualdad de (6.5) tiene un significado geométrico bastante más interesante que la propia fórmula en si. Se trata del siguiente Teorema.

Teorema 6.22 *Sea F una cúbica no singular y sean $L'_1, L'_2, L'_3, L''_1, L''_2, L''_3$ rectas, con $L'_i \neq L''_j$ para todo i, j. Consideremos los 9 puntos de las intersecciones $V(L'_i) \cap V(L''_j)$ y supongamos que 8 de estos nueve puntos están en $V(F)$. Además suponemos que si $\{P\} = V(L'_i) \cap V(L''_j) = V(L'_i) \cap V(L''_k)$, con $j \neq k$, entonces $L'_i = T_P(F)$ y la misma propiedad se verifica intercambiando los papeles de las L'_i y las L''_i. Entonces el noveno punto también pertenece a $V(F)$.*

Explicar la relación entre (6.5) y el Teorema 6.22.

(15) Verificar que el cambio (6.15) transforma la ecuación (6.4) en la ecuación (6.10) y el cambio (6.16) transforma la ecuación (6.10) en la ecuación (6.11). Verificar también las fórmulas (6.13) y (6.14).

(16) Supongamos que K tiene característica 2 y que F es una curva elíptica sobre K con alguna de las dos formas simplificadas del Problema 6.7.17. Obtener fórmulas para el producto cuerda-tangente en este caso.

(17) Sea K un cuerpo de característica 2.

(a) Demostrar que una curva proyectiva con ecuación afín de la forma $y^2 = P(X)$, con P un polinomio de grado ≥ 3 en $K[X]$ es singular.

(b) Demostrar que toda curva elíptica sobre K es equivalente a una curva dada por una de las dos siguientes ecuaciones

$$Y^2 + XY = X^3 + a_2X^2 + a_6, \quad Y^2 + a_3Y = X^3 + a_4X + a_6.$$

(c) Demostrar que una cúbica cuya curva afín venga dada por la ecuación $Y^2 + XY = X^3 + a_2X^2 + a_6$ es no singular si y solo si $a_6 \neq 0$.

(d) Demostrar que una cúbica cuya curva afín venga dada por la ecuación $Y^2 + a_3Y = X^3 + a_4X + a_6$ es no singular si y solo si $a_3 \neq 0$.

(18) Calcular el discriminante de un polinomio mónico de grado 3 en términos de sus coeficientes.

(19) Demostrar que el único punto de intersección de una curva de Weierstrass con la recta del infinito es $[0, 1, 0]$ y que es un punto no singular.

(20) Sea F una curva de Weierstrass singular. Demostrar que el número de puntos singulares de $F(\overline{K})$ es dos si la característica es distinta de dos y uno en caso contrario.

Index