UNIVERSIDAD AUTÓNOMA METROPOLITANA - IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

FACTORIZACIÓN DE ENTEROS

Tesis que presenta
Leonel Sergio Carrasco Pérez
Para obtener el grado de
Maestro en Ciencias (Matemáticas Aplicadas e Industriales)

Asesor: **Dr. José Noé Gutiérrez Herrera**

Jurado Calificador:

Presidente:	**Dra. Martha Rzedowski Calderón**
Secretario:	**Dr. Mario Pineda Ruelas**
Vocal:	**Dr. José Noé Gutiérrez Herrera**

México, D. F. 20 de julio de 2012

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Casa abierta al tiempo

Unidad Iztapalapa

División de Ciencias Básicas e Ingeniería

Maestría en Ciencias Matemáticas Aplicadas e Industriales

Factorización de enteros

Por:

LEONEL SERGIO CARRASCO PÉREZ

TESIS

Para obtener el grado de:

MAESTRO EN CIENCIAS MATEMÁTICAS APLICADAS E INDUSTRIALES

Dirigida por:

Dr. José Noé Gutiérrez Herrera

México, Distrito Federal **20 de julio de 2012**

AGRADADECIMIENTOS

A mi familia por su apoyo incondicional.

A mi asesor Dr. José Noé Gutiérrez Herrera y sinodales: Dra. Martha Rzedowski Calderón y Dr. Mario Pineda Ruelas por su paciencia y tiempo dedicado, por sus comentarios y observaciones que enriquecieron este trabajo.

A mis profesores por ser parte de mi formación académica.

A Dr. Horacio Tapia Recillas por permitirme utilizar el equipo de cómputo de laboratorio de criptografía.

A Josué por recomendarme trabajar con GP/PARI y por su amistad.

A mis amigos y compañeros por todos los buenos momentos que hicieron mi estancia en el posgrado mas agradable.

¡Muchas gracias!

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el primer capítulo se da la definición de criptosistema y se ilustra tal concepto con el criptosistema RSA. El capítulo concluye mostrando que un mensaje cifrado utilizando RSA puede ser fácilmente recuperado por cualquiera que logre factorizar el módulo empleado. Se muestra además que la seguridad del criptosistema RSA se basa en la dificultad computacional de factorizar un número dado. En el segundo capítulo se revisan algunos de los más importantes métodos para factorizar enteros, con objeto de tomarlos en consideración a la hora de crear un juego de llaves en el criptosistema RSA. En el tercer capítulo se implementan algunos de los métodos analizados en el segundo capítulo, la implementación se realizó inicialmente utilizando el software de Wolfram Mathematica 8.0 pero el tiempo requerido en la factorización se mejoró al realizar la implementación en GP/PARI CALCULATOR Versión 2.5.0; el equipo utilizado para dicha implementación es el que se encuentra en el Laboratorio de Criptografía del Departamento de Matemáticas, de la Universidad Autónoma Metropolitana Unidad Iztapalapa.

INTRODUCCIÓN

Desde la época de los antiguos sabios griegos dos problemas en Teoría de Números mantuvieron ocupados a muchos matemáticos: determinar si un entero dado es primo y expresar un entero dado como producto de primos. En el año 2002 el primer problema fue resuelto [7], al publicarse un algoritmo capaz de decidir si el entero que se proporciona como entrada es primo. Actualmente se utiliza un algoritmo que nos dice con una probabilidad cercana a uno si un número es compuesto o bien si es primo. La razón de seguir utilizando este algoritmo probabilístico es que arroja la respuesta mucho más rápido que el algoritmo determinista. El par de problemas mencionados es importante en Criptografía, en particular en el criptosistema conocido como RSA, que fue publicado en 1978 y enseguida fue patentado; la patente expiraba en el año 2000, sin embargo fue liberada dos semanas antes de que esto ocurriera. Éste fue el primer sistema de cifrado en el que cada usuario tiene dos llaves, una pública y otra secreta, para la comunicación segura a través de un canal inseguro, como lo es hoy el internet. Aún hoy en día el RSA es el estándar de comunicación electrónica segura más utilizado, y puede ser usado incluso para la firma digital de documentos.

En este trabajo se dan a conocer los términos utilizados en criptografía así como una descripción explícita del criptosistema RSA: cómo es la creación del juego de claves (pública/privada) que utiliza el criptosistema. También se da un ejemplo de como cifrar un texto con dicho criptosistema haciendo notar que si se realiza un ataque factorizando el módulo utilizado por el criptosistema, el atacante podrá descifrar los textos cifrados con tal criptosistema. Debido a esto se revisan algunos de los métodos especializados en la factorización de números enteros más utilizados en la actualidad. Se implementan algunos de los métodos discutidos en el presente trabajo, ejecutándolos con algunos números propuestos con menos de 25 dígitos, dando un referente del rendimiento que se obtiene

dentro de los métodos implementados, sin embargo esto no quiere decir que el método con el mejor rendimiento será igual de eficiente para números con 25 dígitos o más.

CAPÍTULO I

CRIPTOGRAFÍA

La palabra **criptografía** proviene de la unión de los términos griegos $\kappa\rho\upsilon\pi\tau\omega$ kripto (oculto) y $\gamma\rho\alpha\varphi\omega\varsigma$ graphos (escribir), y su definición es: "escritura oculta".
Los orígenes de la criptografía se hunden en las profundidades de la historia. En todas las épocas ha habido necesidad de comunicar información de forma secreta. Los primeros en usar un método de comunicación secreta sistemático fueron los antiguos habitantes de Esparta. No obstante, si una figura histórica se ha asociado a los orígenes de la criptografía ésta es la de Julio César.
Obviamente la criptografía hace años que dejó de ser un arte para convertirse en una técnica, o más bien un conglomerado de técnicas, que tratan sobre la protección-ocultamiento frente a observadores no autorizados de la información.
En este capítulo se da la definición de un criptosistema así como los tipos en los que se divide, los tipos de criptoanálisis que puede haber, y una descripción explícita del criptosistema RSA. Por último terminamos dando la pauta para realizar un criptoanálisis al criptosistema RSA al realizar la factorización del módulo utilizado.

§1. Criptosistema

Un **criptosistema** es una quíntupla $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, donde:

- $\mathcal{P}$ representa el conjunto de mensajes sin cifrar denominado **texto claro** que pueden ser enviados.
- $\mathcal{C}$ representa el conjunto de todos los posibles **mensajes cifrados**.
- $\mathcal{K}$ representa el conjunto de **claves** que se pueden emplear en el criptosistema.

- $\mathcal{E}$ es el conjunto de transformaciones de **cifrado** o familia de funciones que se aplica a cada elemento de $\mathcal{P}$ para obtener un elemento de $\mathcal{C}$.
- $\mathcal{D}$ es el conjunto de transformaciones de **descifrado**, análogo a $\mathcal{E}$.

Para cada $k \in \mathcal{K}$ hay una función de cifrado $E_k \in \mathcal{E}$ y su correspondiente función de descifrado $D_k \in \mathcal{D}$:

$$E_k : \mathcal{P} \to C \qquad\qquad D_k : C \to \mathcal{P}.$$

Además se verifica la propiedad

$$D_k(E_k(x)) = x, \forall x \in \mathcal{P},$$

es decir, que si un mensaje $x \in \mathcal{P}$ es cifrado como $E_k(x)$ y luego descifrado empleando D_k, obtenemos de nuevo x.

Existen dos tipos fundamentales de criptosistemas:
Criptosistemas simétricos o de clave privada y criptosistemas asimétricos o de clave pública.
En las siguientes secciones se mencionará una descripción más detallada de estos tipos de criptosistema.

§1.1. Criptosistemas simétricos o de clave privada

Los criptosistemas simétricos o de clave privada son aquellos que emplean la misma clave $k \in \mathcal{K}$ tanto para cifrar como para descifrar. Este tipo de criptosistemas presentan el inconveniente de que al ser empleados en comunicaciones, la clave k debe ser de conocimiento común tanto del emisor como del receptor, lo cual lleva a preguntarnos cómo transmitir k de forma segura.
El problema principal que presentan estos tipos de cifrado es la gran cantidad de claves necesarias, pues cada pareja de usuarios que deseen comunicarse requiere de una clave.

Algoritmo de Julio César

Se cuenta que Julio César enviaba los mensajes a sus generales sustituyendo cada letra del alfabeto por la correspondiente tres posiciones más avanzadas:

$Alfabeto\ claro$	a	b	c	d	$\ldots$	x	y	z
$Alfabeto\ cifrado$	D	E	F	G	$\ldots$	A	B	C

Ejemplo:

$Texto\ claro$	$este$	es	un	$mensaje$	$cifrado.$
$Texto\ cifrado$	$HVWH$	HV	XQ	$PHQVDMH$	$FLIUDGR.$

Este criptosistema nos presenta los elementos básicos del proceso de cifrado de un texto:

1. Texto claro: mensaje a cifrar, está escrito en letras minúsculas.
2. Texto cifrado: el mensaje a enviar, escrito en letras mayúsculas.
3. Clave: En este caso es el número 3.
4. Cifrado: la transformación al sumar 3 al número de orden de las letras del texto claro y cambiar el número con la letra correspondiente.
5. Descifrado: basta con restar 3 al número de orden de las letras del texto cifrado y sustituir el número con la letra correspondiente.

§1.2. Criptosistemas asimétricos o de clave pública

Los protocolos asimétricos explotan un punto de vista distinto al de los mencionados en la sección anterior. Estos se introdujeron en 1976 a raíz de los trabajos de W. Diffie y M. Hellman [3] que pretenden eliminar el difícil problema de la transmisión de claves.
Los criptosistemas asimétricos emplean una doble clave (k_p, k_P) donde k_p se conoce como clave privada y k_P se conoce como clave pública. Una de ellas sirve para la transformación E_{k_P} de cifrado y la otra para la transformación D_{k_p} de descifrado. En muchos casos son intercambiables, esto es, si empleamos una para cifrar, la otra sirve para descifrar y viceversa. Estos criptosistemas deben cumplir que el conocer la clave pública k_P no permita calcular la clave privada k_p. Los sistemas de cifrado ofrecen un abanico superior de posibilidades, pudiendo emplearse para establecer comunicaciones seguras por canales inseguros puesto que únicamente viaja por el canal la clave pública, o bien para llevar a cabo autenticaciones.
Por ejemplo, es fácil multiplicar dos números primos distintos p, q y obtener el número $N = p\,q$. No obstante, el proceso inverso, es decir, dado N lo suficientemente grande encontrar sus factores primos p y q es mucho más difícil. En la dificultad de factorizar un número de gran magnitud reside la seguridad de algunos sistemas de clave pública. La seguridad es meramente computacional, dado que el tiempo y recursos que hay que invertir para poder deducir la clave privada a partir de la pública son demasiados, pero matemáticamente es un problema soluble.

§2. Criptoanálisis

Denominamos **criptoanálisis** al ataque a un criptosistema. Hay que considerar distintos tipos de criptoanálisis, básicamente:

- Ataque de texto cifrado. El criptoanalista sólo conoce el texto cifrado y quiere conseguir el texto claro o la clave.

- Ataque con texto claro conocido. Se conoce uno o varios textos en claro con sus correspondientes textos cifrados y se desea determinar la clave.

§3. El criptosistema RSA con clave pública

En 1977 Ronald Rivest, Adi Shamir y Leonard Adleman crearon el denominado sistema RSA. Este criptosistema, el primero de clave pública, uno de los más populares hoy en día por su uso en Internet, está basado en congruencias, por lo que se recordarán algunos conceptos sobre el tema.
Sean a, b y n números enteros, $n \geq 2$, si b es el residuo cuando a es dividido por n, entonces $a - b$ es precisamente divisible por n. Si una diferencia de dos enteros a y b (cada uno de los cuales puede ser positivo o negativo) es precisamente divisible por un entero positivo n, entonces decimos que a y b son **congruentes** módulo n y lo denotamos como,

$$a \equiv b \pmod{n}. \tag{1.1}$$

De forma equivalente podemos decir que el residuo es cero cuando $a - b$ es dividido por n, es decir, la Congruencia (1.1) puede ser escrita como

$$a - b \equiv 0 \pmod{n}. \tag{1.2}$$

Pero si $a - b$ es exactamente divisible por n, entonces $a - b - n$ es divisible por n, por lo tanto

$$a \equiv b + n \pmod{n}. \tag{1.3}$$

Otro aspecto importante en el criptosistema RSA son los números primos ya que constituyen la pieza básica en la construcción de éste. Un número primo se define de la siguiente manera.

Definición I.1. Un número entero es **primo** si tiene exactamente cuatro divisores.

Son primos -7, 13 y 17, pero no 15, ya que los divisores de este número son -15, -5, -3, -1, 1, 3, 5 y 15. Limitaremos nuestra atención a números primos positivos.

Quien desee crear un juego de claves (pública/privada) en el criptosistema RSA, primero selecciona dos números primos p, q diferentes, lo suficientemente grandes[†]. Entonces calcula su producto $N = p\,q$. Después evalúa la función de Euler[††] $\phi(N) = (p-1)(q-1)$, y selecciona un número entero positivo e con $1 < e < \phi(N)$ tal que e sea coprimo con $\phi(N)$. Finalmente calcula un número entero d, con $1 < d < \phi(N)$ tal que

[†] En la práctica estos primos son del orden de 10^{200}.
[††] Si $N = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ entonces $\phi(N) = \prod_{i=1}^{k} p_i^{e_i - 1}(p_i - 1)$, de donde $\phi(N) = (p-1)(q-1)$ si $N = pq$.

$$de \equiv 1 \pmod{\phi(N)}. \tag{1.4}$$

Este inverso puede calcularse fácilmente empleando el Algoritmo Extendido de Euclides.

Finalmente se ha creado el juego de claves, la clave pública son los valores N y e, mientras que la clave privada es el valor d.

Ejemplo I.1. Sean $p = 103$ y $q = 199$ dos números primos, con $N = 20497$. Se calcula $\phi(N) = 102 \cdot 198 = 20196$, y si se elige $e = 8207$, este es coprimo con $\phi(N)$. Ya está completa la clave pública, ahora es necesario calcular d, tal que $de \equiv 1 \pmod{\phi(N)}$. Se obtiene $d = 3455$ con lo que se completa la clave privada.

Para obtener los elementos $c \in \mathcal{C}$ se debe convertir el conjunto de textos en claro $\mathcal{P}$ en una sucesión de enteros positivos $x < N$ y proceder a cifrarlos de la siguiente forma

$$c \equiv x^e \pmod{N}. \tag{1.5}$$

Cuando $\mathrm{mcd}(x, N) = 1$ se descifra c, al utilizar la clave privada d, de la siguiente forma:

$$x \equiv c^d \pmod{N}. \tag{1.6}$$

Por la Congruencia (1.4) se tiene que $d\,e = k\,\phi(N) + 1$, entonces podemos verificar lo anterior:

$$\begin{aligned} c^d &\equiv (x^e)^d \pmod{N} \\ &\equiv x^{k\,\phi(N)+1} \pmod{N} \\ &\equiv (x^{\phi(N)})^k x \pmod{N} \\ &\equiv x \pmod{N}. \end{aligned}$$

Por otro lado, si $\mathrm{mcd}(x, N) > 1$, se descifra c con ayuda del Teorema Chino del Resto de la siguiente forma:

- Calcular q', el inverso de $q \pmod{p}$ y p' el inverso de $p \pmod{q}$.
- Calcular $s := c^{d \pmod{p-1}} \pmod{p}$ y $t := c^{d \pmod{q-1}} \pmod{q}$.
- Recuperar el mensaje $x \equiv q\,q'\,s + p\,p'\,t \pmod{N}$.

En el punto tres se obtiene el mensaje x al resolver el sistema de congruencias del punto dos.

Las Congruencias (1.1) y (1.3) implican que la Congruencia (1.5) no determina unívocamente c, a menos que c sea un número positivo menor que N, lo cual garantiza que c sea el residuo, cuando x^e es dividido por N. Similarmente la Congruencia (1.6) no especifica

unívocamente a x sin la condición adicional de que x sea un número positivo menor que N. Por lo que tales condiciones deben ser integradas cuando se calculen c y x.

Algoritmo 1 RSA

Resumen Cifrado de un mensaje $\mathcal{P}$ y descifrado.

1. ***Cifrado:***
   - **a)** Se debe obtener la clave pública (N, e).
   - **b)** Representar el mensaje x como una sucesión de enteros $x_1, x_2, \ldots, x_t$ en el intervalo $[0, N-1]$.
   - **c)** Calcular $c_i \equiv x_i^e \pmod{N}$, $i = 1, 2, \ldots, t$.
   - **d)** Enviar el texto cifrado, $c = c_1, c_2, \ldots, c_t$.
2. ***Descifrado*** Para recuperar el texto claro a partir de c se debe hacer lo siguiente:
   - **a)** Usar la clave privada d y calcular $x_i \equiv c_i^d \pmod{N}$ $1 \leq i \leq t$, para recuperar x.

Ejemplo I.2. Se desea cifrar el siguiente mensaje, con la clave pública $N = 20497$ y $e = 8207$

"Hombres necios que acusáis
a la mujer sin razón,
sin ver que sois la ocasión
de lo mismo que culpáis"
Sor Juana Inés de la Cruz

Lo primero que haremos es eliminar los espacios en blanco y los separamos en bloques de 3, después asignaremos a cada letra su correspondiente número a= 0, b= 1, c= 2, ... y calculamos:

$$\begin{array}{ccccc} \text{Hom} & \to & 7(26)^2 + 14(26) + 12 & = & 5108 \\ \text{bre} & \to & 1(26)^2 + 17(26) + 4 & = & 1122 \\ \text{sne} & \to & 18(26)^2 + 13(26) + 4 & = & 12510 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{array}$$

utilizando la Congruencia (1.5) se calcula:

$$\begin{array}{cccc} 5108^{8207} & \equiv & 5626 & (\text{mód } 20497). \\ 1122^{8207} & \equiv & 13651 & (\text{mód } 20497). \\ 12510^{8207} & \equiv & 16640 & (\text{mód } 20497). \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

así sucesivamente:

5626	13651	16640	1375	4970	3311	2364
11065	3070	4255	8142	11967	3053	3950
14317	5642	11074	4336	17182	2794	12850
10517	6034	11099	17878	15083	5943	13221
9354	7830	15000	13067	13924	1678	

por último para obtener el texto cifrado, cada número es transformado en base 26 y representado por 4 números, sustituido por la correspondiente letra

$$\begin{array}{ccc} 5626 & \to & \{0,8,8,10\}_{26} \\ 13651 & \to & \{0,20,5,1\}_{26} \\ 16640 & \to & \{0,24,16,0\}_{26} \\ \vdots & \vdots & \vdots \end{array}$$

AIIKAUFBAYQAACAXAHJEAEXJADMYAQ
JPAEOCAGHRAMBEARSHAENLAFVYAVER
AIJAAQJYAGKUAZKWAEDMATAGAPONAI
YCAQKXBALQAWIDAIUPATONANVUALPE
AWEYATIPAUPOACMO

Para este ejemplo se hizo un pequeño programa (véase el anexo) con el cual se realizó el cifrado del texto, así como el descifrado.

§4. Un ataque al RSA

Un criptoanálisis al criptosistema RSA se logrará si de alguna forma se conoce cuál es el valor de $\phi(N)$.

- Supongamos que el valor de $\phi(N)$ es conocido y que un intruso logra interceptar algún mensaje, del cual tiene conocimiento que fue cifrado con dicho criptosistema y conjunto de claves. Como los valores de N y e son públicos, el intruso sólo tendría que calcular el valor de d dado por la Congruencia (1.4) para poder descifrar el texto interceptado al utilizar la Congruencia (1.6), y así habrá tenido éxito al realizar un criptoanálisis al criptosistema.
- Por otro lado supongamos que se logra la factorización de N, es decir $N = p\,q$. Ahora, como es conocido el valor de p y q se puede calcular el valor de $\phi(N)$ y así estaríamos en el caso anterior, donde se logró un criptoanálisis al suponer que se conocía el valor de $\phi(N)$.

Pero el problema de calcular el valor de $\phi(N)$ es equivalente a factorizar N. Ya que si se logra la factorización de N se podría calcular fácilmente el valor de $\phi(N)$. Por otro lado si suponemos que se conoce el valor de $\phi(N)$, la factorización de N se lograría al resolver el siguiente sistema de ecuaciones que se forma:

$$N = p\,q. \tag{1.7}$$
$$\phi(N) = (p-1)(q-1). \tag{1.8}$$

De la Ecuación (1.7) se despeja $q = N/p$ y se sustituye en la Ecuación (1.8) con lo que se obtendría una ecuación cuadrática en términos de p

$$p^2 - (N - \phi(N) + 1)p + N = 0.$$

Las raíces de esta ecuación son los valores de p y q con lo cual se logra la factorización de N.

Ejemplo I.3. Supongamos que $\phi(N) = 84754668$ es conocido y que el valor de $N = 84773093$ también lo es. Con esta información se logra obtener la ecuación:

$$p^2 - 18426\,p + 84773093 = 0$$

resolviendo la ecuación se encuentran las dos raíces del sistema con $p = 9539$ y $q = 8887$ las cuales son los factores primos de N. Con lo que se habrá logrado la factorización de N.

Éste fue sólo un pequeño ejemplo, ya que los números que se utilizan en el criptosistema RSA son de gran magnitud. Por ejemplo para la factorización del número conocido como RSA-768, que consta de 768 dígitos binarios, se emplearon 80 procesadores únicamente para seleccionar ciertos polinomios que se utilizarían en la factorización [8]. La selección de los polinomios tardó aproximadamente medio año. Para el proceso de factorización se emplearon varios cientos de computadoras, y la factorización requirió casi dos años, adicionales al medio año en que se seleccionaron los polinomios. Se estima que si solamente se empleara una computadora con procesador AMD a 2.2 GHz con 2 GB de RAM se requerirían aproximadamente 1500 años para la factorización. El grupo que logró factorizar este número estaba formado por investigadores radicados en Suiza, Francia, Japón, Holanda y Alemania, es decir fue un esfuerzo realmente internacional.

CAPÍTULO II

MÉTODOS DE FACTORIZACIÓN

Puesto que la seguridad del criptosistema RSA descansa en el problema de factorización, en este capítulo se revisarán algunos de los métodos más conocidos y sus algoritmos, como medio para establecer una certidumbre "razonable" sobre la confiabilidad del criptosistema. El principal ataque al sistema RSA consiste en intentar factorizar un número N. Por tanto primero analizaremos algunos métodos de factorización con objeto de tomarlos en consideración en la elección de los números primos p y q.

§1. Historia de la factorización

La humanidad a lo largo de la historia ha tratado de comprender los números primos. Recordamos algunos de los principales matemáticos que realizaron aportaciones sobre el tema en tiempos pasados.

EUCLIDES (325-265 a. C.)
Euclides demuestra que hay infinidad de números primos en la proposición número 20 del libro IX de los Elementos: "Ningún conjunto de números primos los incluye a todos". La demostración es muy sencilla:
Supongamos que hay una cantidad finita de números primos $p_1, p_2, ..., p_k$. Consideramos 1 más el producto de todos ellos, es decir, $N = (p_1 \cdot p_2 \cdot \ldots \cdot p_k) + 1$. Este número podría resultar ser un número primo, o bien es divisible por algún primo no considerado. En cualquier caso el primo p resultante satisface $p \neq p_i,\ i = 1, 2, \ldots, k$.

1	**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

Tabla 2.1: Criba de Eratóstenes

ERATÓSTENES (276-195 a. C.)

Cerca del año 200 a. C., el astrónomo Eratóstenes de Cirene ideó un algoritmo para calcular números primos llamado Criba o Tamiz de Eratóstenes: La criba de Eratóstenes es un algoritmo que permite hallar todos los números primos menores que un número natural dado N. Se forma una tabla con todos los números naturales comprendidos entre 2 y N y se van tachando los números que no son primos de la siguiente manera: se toma el 2 y se procede a tachar todos sus múltiplos, continúa el proceso con el siguiente entero que no ha sido tachado y es declarado primo, después se tachan todos sus múltiplos, así sucesivamente. El proceso termina cuando ya no hay números por tachar.

En la Tabla 2.1 se muestran los números primos menores que 100.

FERMAT (1601-1665)

El próximo gran descubrimiento relacionado con la teoría de números primos fue realizado por Fermat muchos siglos más tarde. Pierre de Fermat nació en Francia en 1601 y las matemáticas eran su gran afición. Él demostró que cada número primo p, tal que $p-1$ es divisible entre cuatro, se puede escribir como la suma de dos cuadrados, el número dos también se incluye, ya que $1^2+1^2=2$. Ideó además un nuevo método de factorización de números grandes, y un teorema importante conocido como Pequeño Teorema de Fermat; establece que si p es un número primo, entonces para cualquier entero a primo relativo con p, obtenemos $a^{p-1} \equiv 1$ módulo p.

Fermat mantuvo correspondencia con otros matemáticos de su época, y en particular con el monje Marín Mersenne (1548-1688), que nació en Francia y actuó de intermediario entre algunos matemáticos del siglo XVII. En una de sus cartas a Mersenne, Fermat conjetura que los números 2^N+1 eran siempre primos si N es una potencia de 2. Él había verificado esto para $N=1$, 2, 4, 8 y 16 y sabía que si N no era una potencia de 2, el resultado fallaba. Los números de esta forma son llamados Números de Fermat. Pero 100 años más tarde Euler demostró que $2^{32}+1 = 4294967297$ es divisible por 641 y por tanto no es primo. Los

números de la forma $2^N - 1$ también atrajeron la atención porque es muy fácil demostrar que a menos que N sea primo, este número es compuesto. A menudo éstos son llamados números primos de Mersenne, dado que Mersenne los estudió. No todos los números de la forma $2^N - 1$ con N primo son primos, por ejemplo si $N = 11, 23, 29, 37, \ldots$ entonces $2^N - 1$ no es primo, en particular $2^{23} - 1 = 8388607 = 47 \cdot 178481$.

§2. Factorización por divisiones sucesivas

El procedimiento teóricamente más sencillo para la búsqueda de los factores primos de un entero cualquiera N, consiste en tomar una sucesión de los primeros valores primos y proceder a calcular la división de N entre la sucesión de primos, iniciando con el primero de ellos. Cada vez que se encuentra un primo p que divide a N se inicia el proceso a partir de ese primo, renombrando a $N = N/p$. Si, se llega a $\lfloor\sqrt{N}\rfloor$ y no se ha encontrado ningún primo que divida a N, se declara primo a N y podemos dar por terminado el proceso.

En lo que resta del presente escrito se analizarán varios métodos de factorización más eficientes que el de divisiones sucesivas.

§3. Método ρ de Pollard

El método ρ de Pollard fue inventado por John Pollard en 1975. Es un método efectivo al factorizar números compuestos que tengan factores menores que 10^{12}.
Se toma una función polinomial $f : \mathbb{Z}_N \to \mathbb{Z}_N$ y un valor inicial $x_0 \in \mathbb{Z}_N$[†]. Se procede a construir una sucesión $x_0, x_1, x_2, \ldots$ usando una iteración de la forma $x_{i+1} = f(x_i)$ (mód N), tomando a x_0 aleatoriamente en $(0, N-1)$. Entonces $\{x_0, x_1, \ldots\} \subseteq \mathbb{Z}_N$. Por lo que la sucesión x_i empieza a tomar un comportamiento cíclico. La secuencia aparece "como un círculo del que pende un tallo", es decir, adquiere la apariencia de la letra griega ro (ρ), que es lo que da nombre al método.

Supongamos que ya se ha calculado la sucesión $x_0, x_1, x_2, \ldots$ Si p es un factor primo de N, y además para ciertos índices i, j se cumple

$$\begin{cases} x_i \equiv x_j & (\text{mód } p) \\ x_i \not\equiv x_j & (\text{mód } N) \end{cases}$$

entonces, como $x_i - x_j = kp$, resulta que $\text{mcd}(x_i - x_j, N)$ es un factor no trivial de N.

Claro, no conocemos p, pero conocemos los x_i, así que podemos revelar la existencia de p con el cálculo del $\text{mcd}(x_i - x_j, N)$. En la práctica se requiere comparar, de manera eficiente, los x_i con los x_j (mód N), hasta encontrar la presencia del factor p.

[†] Sea $\mathbb{Z}_N$ el conjunto de residuos módulo N con $N > 1$.

La función polinomial $f(x)$ es un polinomio irreducible, usualmente se utiliza $f(x) = x^2+1$ (mód N) o alguno similar y se inicia con un elemento x_0 elegido al azar en $\mathbb{Z}_N$. La función polinomial $f(x) = x^2 - 1$ (mód N) aunque no es irreducible también es utilizada.

Definición: Si N es impar y compuesto tal que

$$2^{N-1} \equiv 1 \pmod{N},$$

se dice que N es **pseudoprimo**.

Advertencia: Este método nunca se debe ejecutar sin antes haber realizado una prueba de pseudoprimo (ver Apéndice A).

Ejemplo II.1. Supóngase que se desea factorizar $N = 1387$. Para crear una sucesión usamos $f(x) = x^2 - 1$ y $x_0 = 2$. Luego,

$$\begin{aligned} x_0 &= 2 \\ x_{i+1} &= x_i^2 - 1 \pmod{N} \end{aligned}$$

genera la sucesión,

$$\begin{aligned} \{x_0, x_1, x_2, \ldots\} = \{&2, 3, 8, 63, 1194, \\ &1186, 177, 814, 996, 310, 396, 84, 120, 529, 1053, 595, 339, \\ &1186, 177, 814, 996, 310, 396, 84, 120, 529, 1053, 595, 339, \ldots\} \end{aligned}$$

Así, "por inspección" logramos ver que $1186 \not\equiv 8$ (mód N) y calculando el mcd$(1186 - 8, N) = 19$ se verifica que 19 es un factor de 1387. En este caso detectamos directamente un factor primo de N.

§4. Método $p-1$ de Pollard

En 1974 John M. Pollard propuso un nuevo algoritmo para factorizar enteros. Al igual que el método ρ de Pollard es necesario realizar una prueba de pseudoprimo.

Este método encuentra un factor primo p de un número N. Para ello es necesaria una base formada por números primos consecutivos, es decir, $B = \{2, 3, \ldots, q_t\}$, se sugiere que $q_t < 100000$.
Si $q_i^l \leq N$, entonces $l \ln q_i \leq \ln N$ y así $l \leq \lfloor \frac{\ln N}{\ln q_i} \rfloor$. Se define

$$Q = \prod_{q_i \in B} q_i^{\lfloor \frac{\ln N}{\ln q_i} \rfloor}$$

Si p es un factor primo de N tal que los factores primos de $p-1$ se encuentran en la base B, entonces $p-1|Q$, por consecuencia para cualquier l que satisface $\text{mcd}(l,p)=1$ el Pequeño Teorema de Fermat implica que $l^Q \equiv 1 \pmod{p}$. Ahora si $d=\text{mcd}(l^Q-1,N)$ entonces se habrá encontrado un factor no trivial de N. El método se resume en el Algoritmo 2.

Algoritmo 2 $p-1$ de Pollard

La ***entrada*** $N \in \mathbb{N}$.
La ***respuesta*** cualquiera de los divisores no triviales de N o fracaso.

PASO 1 Tomar una base $B=\{2,3,\ldots,q_t\}$ de primos consecutivos.
PASO 2 Seleccionar un número aleatorio a, $2 \leq a \leq N-1$ y calcular $d=\text{mcd}(a,N)$. Si $d \geq 2$, d es un factor de N, entonces **salida** d y Fin.
PASO 3 Para cada q_i hacer lo siguiente:

1. Calcular $l=\lfloor \frac{\ln N}{\ln q_i} \rfloor$ con $i=1,2,\ldots,t$.
2. Calcular $a \equiv a^{q_i^l} \pmod{N}$.

PASO 4 Calcular $d=\text{mcd}(a-1,N)$. Si $d=1$ o $d=N$, entonces el algoritmo fracasó. En otro caso d es un factor de N. Fin.

A continuación se ilustra el método con un pequeño ejemplo.

Ejemplo II.2. Se desea factorizar $N=19048567$.

1. Seleccionar la base $B=\{2,3,5,7,\ldots\}$.
2. Seleccionar un entero $a=3$ y se verifica que $\text{mcd}(3,N)=1$.
3. Como el valor de $\text{mcd}(3,N)=1$ se procede con el algoritmo y se obtiene la Tabla 2.2.

q_i	$l=\lfloor \frac{\ln N}{\ln q_i} \rfloor$	$a \equiv a^{q_i^l} \pmod{N}$
2	24	2293244
3	15	13555889
5	10	16937223
7	8	15214586
11	6	9685355

Tabla 2.2: Los valores de a en el método $p-1$ de Pollard

4. Calculando $\text{mcd}(a-1, N)$, para los valores anteriores de a, se ve que $d = \text{mcd}(9685355-1, N) = 5281$. Se ha encontrado un factor de N.

5. Los dos factores no triviales de N son $p = 5281$ y $q = N/p = 3607$.

§5. Método de Fermat

En el método de Fermat se buscan dos enteros x e y de tal forma que $x^2 - y^2 = N$. Como $N = (x+y)(x-y)$, salvo que $x - y = 1$, tenemos una factorización de N. Por tanto podemos proceder como en el Algoritmo 3.

Algoritmo 3 Fermat

La ***entrada*** es un número entero N.
La ***respuesta*** es p y q.

PASO 1 Iniciamos con $x := \lfloor \sqrt{N} \rfloor + 1$
PASO 2 Si $y^2 = x^2 - N$ entonces devolvemos $p = x + y,\ q = x - y$. En otro caso $x := x + 1$.

El Algoritmo 3 es eficiente si p y q son dos primos cercanos, ya que el primer número seleccionado $x = \lfloor \sqrt{N} \rfloor + 1$ será cercano a p o q. Por tanto es conveniente que p y q no sean de la misma longitud, es decir que no se expresen con la misma cantidad de bits, si son utilizados para generar una clave en el criptosistema RSA.

Ejemplo II.3. Se desea factorizar $N = 3229799$. Primero se calcula $\sqrt{N} \approx 1797.16$ entonces $\lfloor \sqrt{N} \rfloor + 1 = 1798 = x$ luego $1798^2 - N = 3005$ pero $\sqrt{3005} \notin \mathbb{Z}$ por lo que se incrementa el valor de x y realizamos la diferencia $1799^2 - N = 6602$ pero $\sqrt{6602} \notin \mathbb{Z}$ nuevamente incrementamos x y calculamos $1800^2 - N = 10201$ con lo que se obtiene la raíz $\sqrt{10201} = 101$. Por lo tanto $1800^2 - N = 101^2$ y $1800^2 - 101^2 = N$ luego $(1800+101)(1800-101) = N = (1901)(1699)$, se logró la factorización.

Criba. El nombre de **cribar** viene de limpiar el trigo u otra semilla, por medio de la criba, del polvo, tierra y demás impurezas. La idea de utilizar cribas en cuestiones de factorización es antigua, por ejemplo la criba de Eratóstenes. El Algoritmo 3 puede producir un mejor rendimiento al utilizar una criba: si queremos comprobar que $x^2 - N$ es un cuadrado, una forma de eliminar una cantidad considerable de casos es primero reducir módulo un entero m (que debe ser pequeño) y comprobar si $x^2 - N$ es un cuadrado módulo m. Se supone que se ha calculado previamente una tabla con M elementos que son residuos cuadráticos módulo m, con lo que esta comprobación es rápida. Por otro lado la reducción es considerable, si tomamos primos sucesivos $p_1, \ldots, p_r$ menores a un número B, usualmente $B = 10000$, tomado como cota e iteramos sucesivamente, cada una de las etapas disminuye a la mitad aproximadamente los números x que se deben comprobar teniendo como límite máximo de iteraciones a $A = M/2^r$. Por tanto para comprobar una lista que consta de M enteros, sólo

debemos realizar $M/2^r$ comprobaciones. El método anterior se puede programar como en el Algoritmo 4.

Algoritmo 4 Fermat con criba

La ***entrada*** N entero a factorizar, A, B.
La ***respuesta*** un factor de N o fracaso.

Tomar los factores primos $p_1, \ldots, p_r < B$.
Para $i = 1, \ldots, r$ hacer

- $C[i]$ el conjunto de cuadrados módulo p_i
- $C[i] := (C[i] + N \pmod{p_i})$.

Iniciar con $x = \lfloor \sqrt{N} \rfloor + 1$.
Mientras $x < A + \lfloor \sqrt{N} \rfloor$ hacer

- **Para** $i = 1, ..., r$ hacer
  Si $x^2 \pmod{p_i} \notin C[i]$ entonces $x := x + 1$.
- $y = [\sqrt{x^2 - N}]$
- **Si** $y^2 = x^2 - N$ entonces $q = x - y$ es un factor de N, terminar, **salida** un factor de N es q.

Salida Fracaso pues no se logró encontrar un factor de N.

§6. Fracciones continuas

El método de fracciones continuas fue desarrollado por Morrison y Brillhart en 1970 lo que les permitió factorizar el séptimo número de Fermat $F_7 = 2^{2^7} + 1$. Para poder comprender un poco mejor este método de factorización se recordarán algunos conceptos relacionados con el tema.
Cualquier número real x puede representarse por **fracciones continuas** de la forma siguiente:

$$x = a_0 + \cfrac{z_1}{a_1 + \cfrac{z_2}{a_2 + \cfrac{z_3}{a_3 + \ldots}}}$$

Donde $a_0 \in \mathbb{Z}$, $a_i \in \mathbb{N}$, para $i \geq 1$. Si $z_i = 1$ para todo i esto es llamado una **fracción continua simple** y a menudo es escrita como sigue:

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}}$$

se representa como

$$x = [a_0, a_1, a_2, \ldots]$$

El Algoritmo 5, conocido como algoritmo de las fracciones continuas, nos da la pauta para encontrar la representación de una fracción continua simple.

Algoritmo 5 Representación de una fracción continua

La ***entrada***: número real x.
La ***respuesta***: representación de $\sqrt{x}$ como fracción continua $[a_0, a_1, a_2, \ldots]$.

$$\begin{array}{rcl@{\qquad\qquad}rcl} & & & x_0 & = & \sqrt{x} \\ a_0 & = & \lfloor x_0 \rfloor & x_1 & = & \frac{1}{x_0 - a_0} \\ a_1 & = & \lfloor x_1 \rfloor & x_2 & = & \frac{1}{x_1 - a_1} \\ & \vdots & & & \vdots & \\ a_i & = & \lfloor x_i \rfloor & x_{i+1} & = & \frac{1}{x_i - a_i} \\ & \vdots & & & \vdots & \end{array}$$

Nótese que x es racional si y sólo si su fracción continua es finita.
El número racional

$$\frac{m_i}{n_i} = [a_0, \ldots, a_i]$$

se llama el i**-ésimo convergente de** x. Se tiene que x es el límite de la sucesión de racionales m_i/n_i y de hecho proporcionan una aproximación óptima a x en el sentido de que los denominadores son primos relativos [6]. El numerador y el denominador se pueden obtener recursivamente de la siguiente forma:

$$\begin{array}{rcl} \frac{m_0}{n_0} & = & \frac{a_0}{1} \\ \frac{m_1}{n_1} & = & \frac{a_0 a_1 + 1}{a_1} \\ & \vdots & \\ \frac{m_i}{n_i} & = & \frac{a_i m_{i-1} + m_{i-2}}{a_i n_{i-1} + n_{i-2}}, \quad i \geq 2 \end{array}$$

Una vez calculada la fracción continua simple se procede en el algoritmo a calcular el valor de m_i/n_i, el i-ésimo convergente de $\sqrt{N}$ y se calcula $b_i \equiv m_i^2 \pmod N$. De los valores b_i resultantes sólo se tomarán aquellos cuyos factores primos sean menores a una cota C, el valor máximo de la cota usualmente usado es de 10000, claro que esto depende del valor que se desea factorizar pues un número de 20 dígitos puede ser factorizado con una cota no menor a 3000, pero entre mayor sea la cota utilizada el Algoritmo 6 tardará un poco más en obtener la factorización del número.

Algoritmo 6 Fracciones continuas

La ***entrada*** es un número impar N, una cota C.
La ***respuesta*** es p y q.

PASO 1 Iniciamos con: $m_0 = a_0 = \lfloor\sqrt{N}\rfloor$, $x_0 = \sqrt{N} - a_0$ y $b_0 \equiv a_0^2 \pmod N$.
PASO 2 Para $i = 1, \ldots,$ hacer
$a_i = \lfloor 1/x_{i-1}\rfloor, x_i = (1/x_{i-1}) - a_i$.
Si $i = 1$ entonces $m_i = a_0a_1 + 1$ y $b_i \equiv m_i^2 \pmod N$.
En otro caso $m_i = a_im_{i-1} + m_{i-2}$ y $b_i \equiv m_i^2 \pmod N$.
Descartar los b_i que tienen factores primos mayores a C. De los restantes elegimos algunos valores de b_i tales que sus factores primos aparezcan un número par de veces, para encontrar una congruencia $\prod m_i^2 \equiv \prod p_i^2 \pmod N$.

Ejemplo II.4. Se desea factorizar $N = 17873$. La expansión en fracción continua simple de $\sqrt{N}$ es $[133, 1, 2, 4, 2, 3, 1, 2, 1, 2, 3, 3, \ldots]$ usaremos como cota $C = 29$, por lo que la base de factores es $\{-1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}$ incluyendo el -1 ya que al tomar también los valores $-b_i \equiv m_i^2 \pmod N$ se tendrá una mayor posibilidad de factorizar N. Los cálculos necesarios se muestran a continuación.

$[a_0, \ldots, a_i]$	$=$	$\frac{m_i}{n_i}$	b_i	$\equiv$	$m_i^2 \pmod N$	b_i	$=$	$p_1^{r_1} \cdots p_m^{r_m}$
$[133]$	$=$	133	-184	$\equiv$	$133^2 \pmod N$	-184	$=$	$-1 \cdot 2^3 \cdot 23$
$[133,1]$	$=$	134	83	$\equiv$	$134^2 \pmod N$	83	$=$	primo
$[133,1,2]$	$=$	$\frac{401}{3}$	-56	$\equiv$	$401^2 \pmod N$	-56	$=$	$-1 \cdot 2^3 \cdot 7$
$[133,1,2,4]$	$=$	$\frac{1738}{13}$	107	$\equiv$	$1738^2 \pmod N$	107	$=$	primo
$[133,1,2,4,2]$	$=$	$\frac{3877}{29}$	-64	$\equiv$	$3877^2 \pmod N$	-64	$=$	$-1 \cdot 2^6$
$[133,1,2,4,2,3]$	$=$	$\frac{13369}{100}$	161	$\equiv$	$13369^2 \pmod N$	161	$=$	$7 \cdot 23$

$(133 \cdot 401 \cdot 13369)^2 \equiv (-1 \cdot 2^3 \cdot 7 \cdot 23)^2 \pmod N$. Ahora $133 \cdot 401 \cdot 13369 \equiv 1288 \pmod N$ y $-1 \cdot 2^3 \cdot 7 \cdot 23 \equiv 16585 \pmod N$ pero $1288 \equiv -16585 \pmod N$ esto significa que $\text{mcd}(16585 + 1288, N) = N$ y $\text{mcd}(16585 - 1288, N) = 1$. Así que no conseguimos ningún factor. Por lo que continuamos

$[a_0,\ldots,a_i]$	$=$	$\frac{m_i}{n_i}$	b_i	$\equiv$	m_i^2	(mód N)	b_i	$=$	$p_1^{r_1}\cdots p_m^{r_m}$
[133,1,2,4,2,3,1]	$=$	$\frac{17246}{129}$	-77	$\equiv$	17246^2	(mód N)	-77	$=$	$-1\cdot 7\cdot 11$
[133,1,2,4,2,3,1,2]	$=$	$\frac{47861}{358}$	149	$\equiv$	47861^2	(mód N)	149	$=$	primo
[133,1,2,4,2,3,1,2,1]	$=$	$\frac{65107}{487}$	-88	$\equiv$	65107^2	(mód N)	-88	$=$	$-1\cdot 2^3\cdot 11$

$(401\cdot 3877\cdot 17246\cdot 65107)^2 \equiv ((-1)^2\cdot 2^6\cdot 7\cdot 11)^2$ (mód N). Ahora $401\cdot 3877\cdot 17246\cdot 65107 \equiv 7272$ (mód N) y $(-1)^2\cdot 2^6\cdot 7\cdot 11 \equiv 4928$ (mód N). Tenemos $7272-4928=2344$, al calcular $\mathrm{mcd}(2344,N)=293$ se encuentra un factor no trivial de N y el otro factor es $N/293=61$.

Para efectos de la implementación se realizará la construcción de vectores como se mostrará posteriormente en el método de Dixon, estos vectores facilitarán encontrar la congruencia $\prod m_i^2 \equiv \prod p_i^2$ (mód N) que se utilizará para factorizar el número N como se ha mostrado en el ejemplo anterior.

§7. Criba lineal de Schroeppel

Richard Schroeppel en 1977 ofreció un método de búsqueda de congruencias $u^2 \equiv v^2$ (mód N). La aportación principal de Schroeppel es una ingeniosa forma de reemplazar el tiempo invertido en las pruebas de división por otro, en principio más breve: el tiempo necesario para realizar una criba como la de Erastóstenes. Su idea era encontrar otros modos de producir residuos cercanos a $\sqrt{N}$ que tuviesen la ventaja de que pudieran factorizarse colectivamente mediante la criba.

El algoritmo de Schroeppel es el siguiente. Sea $k=\lfloor\sqrt{N}\rfloor$ y considérese la función:

$$S(a,b)=(k+a)(k+b)-N,$$

donde los valores absolutos de $|a|$ y $|b|$ son menores que N. Tomar valores de $S(a_i,b_i)$, tales que

$$\prod_{i=1}^{k} S(a_i,b_i)=u^2.$$

Esto se puede realizar tomando un valor fijo $a=a_0$ y variar b sobre enteros consecutivos. Además, se debe lograr que cada valor distinto de $a_1,...,a_k,b_1,...,b_k$ sea asumido un número par de veces. Así entonces

$$\prod_{i=1}^{k}(k+a_i)(k+b_i)=v^2.$$

Por tanto, si podemos encontrar colecciones de pares a_i, b_i, entonces podemos encontrar enteros u y v tales que $u^2 \equiv v^2 \pmod N$. Por lo que sólo tendremos que calcular $\mathrm{mcd}(u-v, N) = p$ para encontrar un factor no trivial de N.

Ejemplo II.5. Se desea factorizar $N = 3229799$. Tomando los valores de $a_1, ..., a_k$ y $b_1, ..., b_k$ de 1 hasta $\lfloor \sqrt{N} \rfloor = 1797 = k$. El proceso terminó en $a_k = 428$, calculando

$$\begin{aligned} u &= \sqrt{\prod_{i=1}^{k} S(a_i, b_i)} \pmod N \\ &= 1671816 \end{aligned}$$

y luego el valor de v

$$\begin{aligned} v &= \sqrt{\prod_{i=1}^{k} (k+a_i)(k+b_i)} \pmod N \\ &= 1557983. \end{aligned}$$

Lo siguiente es calcular $p = \mathrm{mcd}(u-v, N) = 1699$ el cual es un factor primo de N. El otro factor es $N/p = 1901$.

§8. Curvas elípticas

Lenstra en 1987 inventó un nuevo método para factorizar enteros utilizando curvas elípticas. Una curva elíptica sobre un campo K es una curva algebraica que viene dada por una ecuación del tipo

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in K, \Delta \neq 0$$

denominada **ecuación general de Weierstrass** [9], donde Δ es el discriminante y está definido como:

$$\begin{aligned} \Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2. \end{aligned}$$

Definición II.1. Para el campo K con identidad multiplicativa 1, la **característica**, denotada car(K) se define como el número entero positivo más pequeño r tal que:

$$\underbrace{1+1+\cdots+1}_{r\text{-veces}}=0.$$

En consecuencia de la definición los campos $\mathbb{Q}$ (Racionales), $\mathbb{R}$ (Reales), $\mathbb{C}$ (Números Complejos) tienen característica 0. Para un número primo p, el campo finito $\mathbb{F}_{p^n}$ tiene característica p.

Si la característica de un campo $\mathbb{F}_q$ es distinta de 2 y 3, usando transformaciones lineales de las variables, la ecuación de la curva se puede expresar como

$$y^2=x^3+ax+b \;\; a,b\in\mathbb{F}_q$$

denominada **ecuación reducida de Weierstrass**, con discriminante $\Delta=-16(4a^3+27b^2)\neq 0$, para que la curva sea **no singular**, es decir, no hay puntos en la curva que tengan dos o más rectas tangentes distintas [9].
Si E es una curva elíptica sobre el campo $\mathbb{F}_q$, denotaremos por $E(\mathbb{F}_q)$ el conjunto de puntos $P(x,y)\in\mathbb{F}_q\times\mathbb{F}_q$ que satisfacen la ecuación de la curva, unión con el punto al infinito O, es decir

$$E(\mathbb{F}_q)=\{(x,y)\in\mathbb{F}_q^2|\, y^2=x^3+ax+b\}\bigcup\{O\}$$

Si el campo es $\mathbb{R}$, la gráfica de una curva elíptica que se obtendría sería similar a una de las que se encuentran en la Figura 2.1.

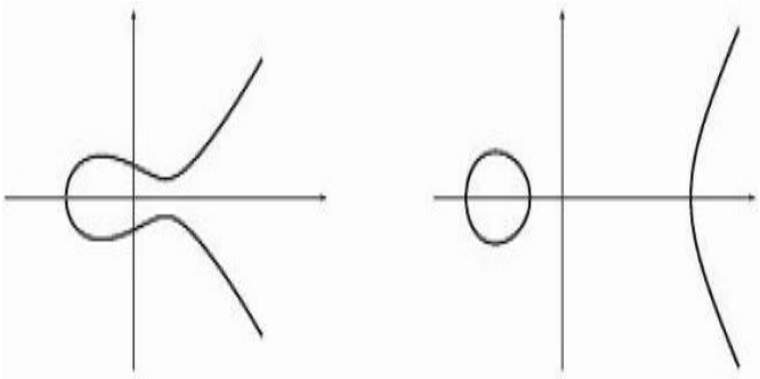

Figura 2.1: Curvas $y^2=x^3-3x+3$ y $y^2=x^3-13x-12$

Para reducir la notación, al referirnos a $E(\mathbb{F}_q)$ sólo escribiremos E de ahora en adelante. Como $\mathbb{F}_q$ es un campo finito con q elementos, obviamente E tiene un número finito de puntos. Por ejemplo, el conjunto de puntos de la curva $E:\;\; y^2=x^3+x+1$ sobre $\mathbb{F}_7$ es

$$E = \{(0,1),(0,6),(2,2),(2,5),O\}.$$

Estructura de grupo. La propiedad fundamental que hace a las curvas elípticas interesantes para la factorización es que tienen una estructura de grupo peculiar. Definimos la operació de grupo como sigue: el inverso aditivo de un punto $P = (u,v) \in E$ es $-P = (u,-v)$, en el caso del plano real, es la imagen reflejada sobre el eje x, y para el punto al infinito tenemos que su inverso será él mismo $-O = O$. Cuando interceptamos una línea a través de P y Q con E, conseguimos tres puntos $\{P,Q,S\}$, como se muestra en la Figura 2.2. Por lo que se define

$$R = P + Q = -S.$$

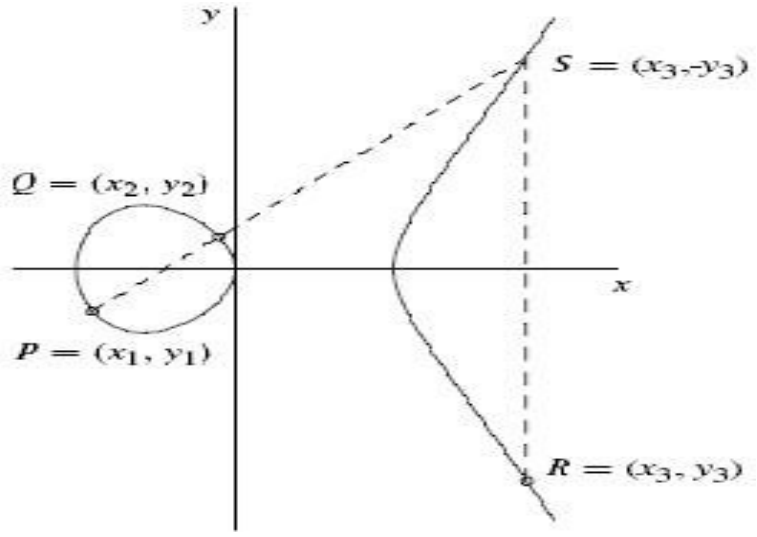

Figura 2.2: Suma de los puntos P y Q sobre una curva elíptica, donde $P + Q = R$ es el inverso aditivo de S

Los casos especiales son:

(i) Si $Q = P$ y tomamos la línea tangente a P. La tangente está siempre bien definida.

(ii) Si $Q = O$ y tomamos la línea vertical a través de P:

$$P + O = -(-P) = P.$$

(iii) Si $Q = -P$ y tomamos otra vez la línea vertical a través de P y Q, obtenemos

$$P + (-P) = -O = O.$$

Esto hace que E sea un grupo abeliano. El segundo caso muestra que O es el elemento neutro de E, y el tercer caso dice que el inverso de un punto P es su negativo $-P$. Entonces se puede definir el punto $k \cdot P$, con $k \in \mathbb{Z}$, como

$$k \cdot P = \begin{cases} \overbrace{P + \ldots + P}^{k}, & \text{si } k > 0 \\ O, & \text{si } k = 0 \\ \overbrace{(-P) + \ldots + (-P)}^{-k}, & \text{si } k < 0. \end{cases}$$

Derivamos ahora la expresión racional para la suma en una curva elíptica E. Supongamos que $P = (x_1, y_1)$, $Q = (x_2, y_2) \in E$ con $x_1 \neq x_2$. Sea $R = (x_3, y_3) = P + Q \in E \backslash \{O\}$. La línea que atraviesa P y Q tiene por ecuación $y = \alpha x + \beta$, donde $\alpha = \dfrac{y_2 - y_1}{x_2 - x_1}$ y $\beta = y_1 - \alpha x_1$. Sea $S = (x_3, -y_3)$ la intersección de la línea con la curva. Entonces $(\alpha x_3 + \beta)^2 = x_3^3 + a x_3 + b$. Ya que x_1, x_2, x_3 son raíces de la ecuación cúbica $(x^3 + ax + b) - (\alpha x + \beta)^2 = 0$, tenemos $x_1 + x_2 + x_3 = \alpha^2$ y por consiguiente:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \qquad y_3 = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3). \tag{2.1}$$

Así, encontramos los coeficientes de la suma de dos puntos distintos. Notamos que estas fórmulas no usan explícitamente los coeficientes que aparecen en la ecuación de E, pues sólo están determinadas por P, Q. De forma similar para obtener un punto R cuando $Q = P$ es decir ($R = 2P$, $x_1 = x_2$ y $y_1 = y_2$), si $y_1 \neq 0$, se tiene:

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1, \qquad y_3 = -y_1 + \frac{3x_1^2 + a}{2y_1} \cdot (x_1 - x_3) \tag{2.2}$$

y $2P = O$ si $y_1 = 0$.

Ahora se utilizarán las mismas fórmulas para la suma de puntos en curvas elípticas, pero sobre los enteros módulo N. Obviamente no siempre será posible encontrar inversos, ya que podrían compartir un factor primo con N pero esto ayudará a factorizar N, como se muestra en el Algoritmo 7. Éste realiza una pequeña prueba pues calcula $\text{mcd}(N, 6)$ para conocer si 2 o 3 son factores de N, el número máximo de iteraciones usualmente usado en el Algoritmo 7 es $B = 10000$.

Algoritmo 7 Curva elíptica

La ***entrada:*** $N \in \mathbb{N}$ compuesto, con $\text{mcd}(N, 6) = 1$, y B para número máximo de iteraciones.
La ***respuesta:*** cualquiera de los divisores no triviales de N o fracaso.

PASO 1 Escoger aleatoriamente (E, P) donde E es una curva elíptica $y^2 = x^3 + ax + b$ sobre $\mathbb{Z}_N$ y $P(x, y) \in E(\mathbb{Z}_N)$.
PASO 2 Hacer $a = 2$.
PASO 3 Mientras $a \leq B$ hacer

- Calcular el punto $P_a \in \mathbb{Z}_N$, que se realiza mediante las sumas repetidas:
  El punto $P_a(x_a, y_a) = P_1(x_1, y_1) + P_{a-1}(x_{a-1}, y_{a-1}) \pmod N$, utilizando la expresión

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2 \pmod N,\ \lambda(x_1 - x_3) - y_1 \pmod N)$$

  donde

$$\lambda = \begin{cases} \dfrac{d_1}{d_2} = \dfrac{3x_1^2 + a}{2y_1} \pmod N, & \text{si } P_1 = P_2 \\ \dfrac{d_1}{d_2} = \dfrac{y_2 - y_1}{x_2 - x_1} \pmod N, & \text{en otro caso.} \end{cases}$$

- Si $c = O$ o si no es posible calcular d_2 entonces, calcular $p = \text{mcd}(d_2, N)$, si $1 < p < N$ entonces **salida** p y terminar el proceso.
- En otro caso incrementar en una unidad el valor de a.

PASO 4 Fracaso, el número de iteraciones no fue suficiente para encontrar un factor de N.

Ejemplo II.6. Utilizamos la curva elíptica $y^2 = x^3 + x + 1$ sobre $\mathbb{Z}_{35}$ para factorizar 35. Tomemos $B = 11$ y el punto $P = (0, 1)$, notemos que pertenece a la curva. Ahora se procede a calcular los múltiplos enteros de P:

	(mód 35)
P	$(0, 1)$
$2P$	$(9, 12)$
$3P$	$(2, 16)$
$4P$	$(28, 34)$
$5P$	

Veamos qué es lo que pasa con $5P$, cuando intentamos calcular $5P$, el calcular $P + 4P$ (mód 35) involucra la expresión $(y_2 - y_1)/(x_2 - x_1)$ y el denominador es $x_2 - x_1 = 28 - 0$

el cual no es invertible módulo 35 ya que $\text{mcd}(28, 35) = 7$. Así hemos encontrado un factor no trivial de 35, por lo que termina el proceso.

§9. Método de Dixon

El método de John D. Dixon fue publicado en 1981 y se basa en una idea muy simple. Si podemos encontrar $x, y \in \mathbb{Z}$ con $x \not\equiv \pm y \pmod{N}$ tales que $x^2 \equiv y^2 \pmod{N}$ entonces $N|(x-y)(x+y)$ pero N no es divisible por $(x-y)$ ni $(x+y)$, por consiguiente $\text{mcd}(x+y,\ N)$ es un factor no trivial de N (lo mismo ocurre con $\text{mcd}(x-y,\ N)$). Como un ejemplo de esto tenemos que $10^2 \equiv 32^2 \pmod{77}$ calculando $\text{mcd}(10+32, 77) = 7$ por lo que 7 es un factor de 77.

El método usa una base $B = \{p_1, ..., p_b\}$ que es un pequeño conjunto de primos sucesivos empezando con el 2. Primero se eligen aleatoriamente algunos valores x_i y se calculan $z_i \equiv x_i^2 \pmod{N}$. Cada valor de z_i es factorizado de la siguiente forma:

$$z_i = p_1^{\alpha_{1\,i}}\, p_2^{\alpha_{2\,i}} \cdots p_b^{\alpha_{b\,i}}.$$

Se eliminan aquellos valores de z_i tales que en su factorización los factores no pertenezcan a los valores en la base B. Para cada i, se considera el vector

$$a_i = (\alpha_{1\,i} \pmod 2, ..., \alpha_{b\,i} \pmod 2) \in (\mathbb{Z}_2)^b.$$

Procuramos encontrar un subconjunto de a_k tal que la suma $\pmod 2$ sea el vector $(0, ..., 0)$, pero si no se logra encontrar tal subconjunto entonces se toma una base B más grande y se vuelve a realizar el procedimiento, una vez que se encontró dicho subconjunto se podrá calcular

$$\prod z_k \equiv \prod x_k^2 \pmod{N}$$

Esto origina la congruencia deseada $x^2 \equiv y^2 \pmod{N}$, que (esperamos) nos llevará a la factorización de N con lo que tenemos un 50 % de probabilidad de encontrar un factor no trivial de N. Ésta es una técnica probabilística de factorización, por lo que no tenemos ninguna garantía de que logremos encontrar algún factor de N, por lo menos en un período determinado de tiempo. Este método es utilizado para factorizar números con menos de 25 digitos.

Ejemplo II.7. Sea $N = 1577078441$ y elijamos $B = \{2, 3, 5, 7, 11, 13\}$. Consideremos las tres congruencias.

$$\begin{aligned} 8340934156^2 &\equiv 3 \cdot 7 \pmod{N} \\ 12044942944^2 &\equiv 2 \cdot 7 \cdot 13 \pmod{N} \\ 2773700011^2 &\equiv 2 \cdot 3 \cdot 13 \pmod{N} \end{aligned}$$

Sean los vectores a_1, a_2, a_3 como sigue

$$\begin{aligned} a_1 &= (0,1,0,1,0,0) \\ a_2 &= (1,0,0,1,0,1) \\ a_3 &= (1,1,0,0,0,1) \end{aligned}$$

es fácil de ver que $a_1 + a_2 + a_3 = (0,0,0,0,0,0) \pmod 2$, si tomamos el producto de estas tres congruencias obtenemos

$$9503435785^2 \equiv 546^2 \pmod N$$

entonces utilizando el algoritmo de Euclides al calcular

$$\mathrm{mcd}(9503435785 - 546, 1577078441) = 115759$$

se encuentra un factor no trivial de N.

En la práctica una manera de escoger los x_i es tomarlos de la forma $i + \lfloor \sqrt{k\,N} \rfloor$, con $i = 0, 1, 2, \ldots$ y $k = 0, 1, 2, \ldots$. Estos enteros tienden a ser pequeños cuando el cuadrado se reduce módulo N por esto hay una mayor probabilidad de factorizar N. Otro truco útil es utilizar enteros de la forma $x_i = \lfloor \sqrt{i\,N} \rfloor$. Si se incluye -1 en la base B, se tendrá una mayor posibilidad de factorizar N, que al no tomarlo en la base B.

Ejemplo II.8. Supongamos que se desea factorizar $N = 2881$ tomando $B = \{-1, 2, 3, 5, 7\}$. Calculamos $\lfloor \sqrt{N} \rfloor = 53$, $\lfloor \sqrt{2N} \rfloor = 75$, $\lfloor \sqrt{3N} \rfloor = 92$, $\lfloor \sqrt{4N} \rfloor = 107$, $\lfloor \sqrt{5N} \rfloor = 120$, $\lfloor \sqrt{6N} \rfloor = 131$, $\lfloor \sqrt{7N} \rfloor = 142$, $\lfloor \sqrt{8N} \rfloor = 151$, $\lfloor \sqrt{9N} \rfloor = 161$, $\lfloor \sqrt{10N} \rfloor = 169$ y $\lfloor \sqrt{11N} \rfloor = 178$.

z_i	$\equiv$	$x_i^2 \pmod N$	z_i	$=$	$p_1^{\alpha_{1\,i}} \cdots p_b^{\alpha_{b\,i}}$
-72	$\equiv$	$53^2 \pmod N$	-72	$=$	$-1 \cdot 2^3 \cdot 3^2$
2744	$\equiv$	$75^2 \pmod N$	2744	$=$	$2^3 \cdot 7^3$
-179	$\equiv$	$92^2 \pmod N$	-179	$=$	$-1 \cdot 179$
-75	$\equiv$	$107^2 \pmod N$	-75	$=$	$-1 \cdot 3 \cdot 5^2$
-5	$\equiv$	$120^2 \pmod N$	-5	$=$	$-1 \cdot 5$
-125	$\equiv$	$131^2 \pmod N$	-125	$=$	$-1 \cdot 5^3$
-3	$\equiv$	$142^2 \pmod N$	-3	$=$	$-1 \cdot 3$
-247	$\equiv$	$151^2 \pmod N$	-247	$=$	$-1 \cdot 13 \cdot 19$
-8	$\equiv$	$161^2 \pmod N$	-8	$=$	$-1 \cdot 2^3$
-249	$\equiv$	$169^2 \pmod N$	-249	$=$	$-1 \cdot 3 \cdot 83$
-7	$\equiv$	$178^2 \pmod N$	-7	$=$	$-1 \cdot 7$

Se eliminan los valores de z_i tales que en su factorización los factores no pertenezca a los valores de la base B, por consiguiente tenemos ocho factorizaciones, las cuales producen los siguientes vectores.

$$\begin{aligned} a_1 &= (1,1,0,0,0) \\ a_2 &= (0,1,0,0,1) \\ a_3 &= (1,0,1,0,0) \\ a_4 &= (1,0,0,1,0) \\ a_5 &= (1,0,0,1,0) \\ a_6 &= (1,0,1,0,0) \\ a_7 &= (1,1,0,0,0) \\ a_8 &= (1,0,0,0,1) \end{aligned}$$

Claramente $a_2+a_3+a_4+a_5+a_6+a_7+a_8 = (0,0,0,0,0) \pmod 2$. Por lo que la congruencia que se obtiene es:

$$(75 \cdot 107 \cdot 120 \cdot 131 \cdot 142 \cdot 161 \cdot 178)^2 \equiv (2^3 \cdot 3 \cdot 5^3 \cdot 7^2)^2 \pmod{2881}$$

simplificando

$$404^2 \equiv 69^2 \pmod{2881}$$

finalmente se calcula

$$\text{mcd}(404 - 69, 2881) = 67$$

se obtiene un factor no trivial de N.

§10. Criba general de campos de números

La criba general de campos de números es un método complejo, comparte muchas similitudes con otros métodos basados en cribar. El nombre de este método es abreviado como (**GNFS**), pues en inglés se denomina"the general number field sieve". Consiste de 5 pasos, los cuales la persona a cargo no puede hacer en paralelo, pero algunos de ellos sí se pueden realizar internamente en paralelo. El Algoritmo 8 describe una idea general de los pasos que se realizan en este método.

Algoritmo 8 Algoritmo GNFS

La ***entrada*** un entero compuesto N.
La ***respuesta*** un factor no trivial p de N.

PASO 1: (Selección de polinomios)
Encontrar un polinomio irreducible $f(x) \in \mathbb{Z}[x]$ y una raíz m módulo N, es decir, $f(m) \equiv 0$ (mód N).
PASO 2: (Las bases de factores)
En la Sección §10.2 se describe como elegir las bases de factores: la base de factores racionales, la base de los factores algebraicos y la base de carácter cuadrático.
PASO 3: (Cribar)
Encontrar los pares de enteros (a, b) con las siguientes propiedades:

- $\text{mcd}(a, b) = 1$
- (a, b) es uniforme sobre la base de factor racional
- (a, b) es uniforme sobre la base del factor algebraico

Un par (a, b) con estas propiedades se llama una relación. El propósito del paso de cribar es coleccionar tantas relaciones como sea posible (por lo menos más que los elementos de todas las bases combinadas). En el paso de cribar resulta un conjunto S de relaciones.
PASO 4: (Álgebra lineal)
Filtrar los resultados de la criba, quitando los duplicados y las relaciones que no contienen un primo presente.
Las relaciones se colocan en conjuntos de relaciones y se construye una matriz sobre $\mathbb{F}_2$.
La matriz es reducida resultando algunas dependencias, es decir, los elementos que conducen a un módulo cuadrado N.
PASO 5: (La raíz cuadrada)
Calcular la raíz cuadrada racional, es decir, y con

$$y^2 = \prod_{(a,b)\in S} (a - bm)$$

Calcular la raíz cuadrada algebraica, es decir, x con

$$x^2 = \prod_{(a,b)\in S} (a - b\alpha)$$

donde α es una raíz de $f(x)$.
p puede encontrarse al calcular $\text{mcd}(N, x - y)$ y $\text{mcd}(N, x + y)$.

§10.1. Selección polinomial

El primer paso del método GNFS es la selección de un polinomio útil y esto no es difícil, pero sí es arduo de encontrar. Un polinomio $f(x)$ se dice que es **bueno** si tiene un **buen rendimiento**. Y hay dos factores principales que influyen en el rendimiento de un polinomio, éstas son las propiedades de tamaño y de raíz.

Definición: Un polinomio $f(x)$ tiene **propiedad de buen tamaño**, si los valores tomados por $f(x)$ son pequeños. Esto puede lograrse mediante la selección de un polinomio con pequeños coeficientes. Cuanto menor sea el tamaño mejor es la propiedad de tamaño.

Definición: Un polinomio $f(x)$ se dice que tiene **buenas propiedades de raíz** si $f(x)$ tiene muchas raíces módulo primos pequeños.

No hay forma directa para elegir un polinomio bueno. El mejor método tiene el objetivo de asegurar un cierto nivel de tamaño y de propiedades de raíz, por lo que se crean los candidatos que son probados y el que tiene el mejor rendimiento es elegido. El problema radica en la búsqueda del polinomio entre los candidatos. En 1999 Brian Anthony Murphy trabajó en su tesis el rendimiento de polinomios [10]. El polinomio $f(x)$ debe tener las siguientes propiedades

1. Es irreducible en $\mathbb{Z}[x]$.
2. Tiene una raíz m módulo N, donde N es el entero a factorizar.

Es fácil crear un polinomio en $\mathbb{Z}_N[x]$ con una raíz deseada m usando una representación en base-m. Esto resulta (en la mayoría de los casos) también en un polinomio irreducible, de lo contrario se ha obtenido la factorización de N, ya que tendríamos que $f(x) = g(x)\,h(x)$ y $f(m) = g(m)\,h(m) = N$.
La construcción de un polinomio $f(x)$ usando la representación de N en base-m puede ser modificado libremente siempre y cuando $f(m) \equiv 0 \pmod{N}$. La representación de N en base-m es

$$N = \sum_{i=0}^{d} a_i m^i$$

con $0 \leq a_i < m$, d está determinado de antemano y está normalmente en el rango de 3 a 6. Un **polinomio** $f(x)$ **de grado** d puede construirse con los a_i como coeficientes, este polinomio tendrá m, como una raíz, es decir:

$$\begin{aligned} f(x) &= a_d x^d + a_{d-1} x^{d-1} + \cdots + a_0 \\ f(m) &\equiv 0 \pmod{N}. \end{aligned}$$

La propiedad de tamaño se puede obtener haciendo los coeficientes más pequeños, por ejemplo, haciendo las sustituciones

$$\begin{aligned} a_i &= a_i - m \\ a_{i+1} &= a_{i+1} + 1 \end{aligned}$$

lo que se traduce en menores a_i conservando $f(m) \equiv 0 \pmod{N}$, es importante para tener a_d y a_{d-1} pequeños.

El mejor rendimiento de un polinomio parece obtenerse mediante polinomios simétricos, pero éstos son mucho más complejos de crear, y por ello se utilizan polinomios no simétricos.

Murphy ha hecho una heurística útil para medir el rendimiento de un polinomio. Se define $\alpha(F)$ como

$$\alpha(F) = \sum_{p \leq B} \left(1 - q_p \frac{p}{p+1}\right) \frac{\ln p}{p-1}$$

donde B es una cota y q_p es el número de raíces distintas de $f \pmod{p}$, $\alpha(F)$ da una estimación muy buena del rendimiento de un polinomio, por lo que se pueden eliminar polinomios malos, y el resto de polinomios se les realiza una pequeña prueba de criba para identificar el mejor rendimiento.

La selección del polinomio consta de los siguientes pasos:

- Identificar un conjunto amplio de polinomios utilizables.
- Con heurística eliminar polinomios obviamente malos del conjunto.
- Hacer pequeños experimentos de criba en el resto de los polinomios y elegir el que tiene el mejor rendimiento.

El Algoritmo 9 describe los detalles para la creación de un polinomio adecuado para su uso en GNFS.

Algoritmo 9 Selección del polinomio no-simétrico

La ***entrada*** un entero compuesto N, entero d.
La ***respuesta*** un polinomio $f(x)$.

PASO 1: Escoger m_0 tal que $\lfloor N^{\frac{1}{d+1}} \rfloor \leq m_0 \leq \lceil N^{\frac{1}{d}} \rceil$.

PASO 2: Escoger el intervalo tal que $\chi_1 < \dfrac{|a_d|}{m_0} < \chi_2$, donde $0 < \chi_1 < \chi_2 < 0.5$.

PASO 3: Definir el intervalo de m tal que $|a_{d-1}|$ es más pequeño. Éste será el intervalo para m_0 donde a_d es cambiado por

$$m_{cambio} = \lfloor \left(\frac{N}{a_d} \right)^{\frac{1}{d}} \rfloor.$$

PASO 4: Para todas las combinaciones utilizables de a_d y $m \in [m_0, m']$, hacer

- Escribir N en base-m, tal que $N = a_d m^d + a_{d-1} m^{d-1} + \cdots + a_0$.
- Escribir el polinomio $f_m(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_0$.
- Checar el rendimiento, calculando $\alpha(f_m)$.

  i Si el rendimiento se satisface agregar $f_m(x)$ al conjunto F' (es el conjunto de polinomios con buen rendimiento).

  ii En otro caso se desecha el polinomio.

PASO 5: Escoger $f(x) \in F'$, haciendo pequeños experimentos de criba.
PASO 6: Salida $f(x)$.

§10.2. Las bases de factores

El algoritmo necesita un dominio bien definido para trabajar, y esto se especifica con **las bases de factores**: la base del factor racional **RFB**, la base del factor algebraico **AFB** y la base del carácter cuadrático **QCB**. Los tamaños de las bases de factores tienen que ser determinadas empíricamente.
La base del factor racional **RFB** está compuesta de todos los primos p_i y $p_i \pmod m$, es decir, por todos los pares $(p, p \pmod m)$, los primos p_i pueden ser tomados como todos los primos menores a una cota w dependiendo del número a factorizar, por ejemplo para factorizar un número de 7 digitos se podrian tomar los primos menores a $w = 60$.

$$\text{RFB} = \{(p_0, p_0 \pmod m), (p_1, p_1 \pmod m), \ldots\}.$$

La base del factor algebraico **AFB** se compone de los pares (p, r) tal que $f(r) \equiv 0 \pmod p$, es decir.

$$\text{AFB} = \{(p_0, r_0), (p_1, r_1), \ldots\}.$$

El tamaño de la base del factor algebraico debe ser 2 o 3 veces más grande que el tamaño de la base del factor racional, una forma de hacer esto es tomar todos los primos menores a $2w$ o $3w$.
La base del carácter cuadrático **QCB** contiene pares (p, r) con las mismas propiedades que los elementos de la base del factor algebraico, pero los p_i son más grandes que los de la base del factor algebraico, es decir.

$$\text{QCB} = \{(p_0, r_0), (p_1, r_1), \ldots\}.$$

El número de elementos en QCB es relativamente pequeño comparado con el número de elementos en RFB y AFB. Es recomendado para factorizar números con menos de 100 dígitos.

§10.3. Criba

El paso de cribar no es teóricamente lo más complejo del algoritmo, pero es el que consume la mayor parte del tiempo porque las iteraciones son sobre un dominio grande con algunos cálculos costosos, como la división y módulo; aunque algunos de éstos calculos pueden evitarse usando logaritmos. En general la optimización del paso de la criba dará una reducción mayor en el tiempo de ejecución del algoritmo.
Es fácil usar una cantidad grande de memoria en este paso, y uno debe ser consciente de esto e intentar reutilizar los arreglos y usar los posibles tipos de los datos más pequeños. El registro de las bases de factores utilizado para la factorización de un número contiene millones de elementos, debido a esto se debe intentar obtener un mejor intercambio en disco/memoria.
El propósito del paso de la criba es encontrar las relaciones útiles, es decir, elementos (a, b) con las propiedades siguientes

- $\text{mcd}(a, b) = 1$.
- (a, b) es **RFB-uniforme**[†].
- (a, b) es **AFB-uniforme**[††].

Para encontrar elementos con estas propiedades se pueden utilizar varios métodos de criba, como la clásica criba lineal o el rápido cribado de retículas.

[†] El par (a, b) se dice que es **RFB-uniforme** si los factores de la norma racional $N(a, b) = a + bm$ son elementos de la RFB.

[††] El par (a, b) se dice que es **AFB-uniforme** si los factores de la norma algebraica $N(a, b) = (-b)^{\deg(f)} f(\frac{a}{b})$ son elementos de la AFB.

La criba lineal se realiza sobre los valores de (a, b), tomando b fijo y variando a en $[-C, C]$ el valor de C depende del número que se desea factorizar por ejemplo para un número de 7 digitos es suficiente con tomar $C = 200$, sin embargo si el valor tomado por C no es lo suficientemente grande como para crear una lista con $\sharp RFB + \sharp AFB + \sharp QCB + 1$ ($\sharp$ representa el número de elementos) elementos como se muestra en el Algoritmo 10, se toma un valor más grande para C. Lo que significa que la norma racional ($N_{racional}(a, b) = a+bm$) es calculada para todos los valores de (a, b) de los cuales sólo se toman aquellos valores de (a, b) que son RFB-uniformes, el mismo procedimiento se hace para la norma algebraica ($N_{algebraica}(a, b) = b^{\deg(f)} f(a/b)$) sobre AFB.

No es necesario comprobar si cada elemento que resulta de la criba lineal sobre las bases de factores, es divisible por cada elemento de RFB y AFB.

Los elementos de la criba lineal que tienen un elemento dado de las bases de factores como un factor, es distribuido a lo largo de la criba lineal por un patrón simple.

- Los elementos (a, b) con norma racional divisible por un elemento (p, r) de RFB son los que tienen a de la forma $a = -bm + kp$ de $k \in \mathbb{Z}$.

- Los elementos (a, b) con norma algebraica divisible por elemento (p, r) de AFB son los que tienen a de la forma $a = -br + kp$ para $k \in \mathbb{Z}$.

Con estas propiedades en mente, podemos tomar cada elemento de la base del factor y quitar su participación de los elementos si es que lo tienen como factor. Esta es la idea del Algoritmo 10 de criba lineal.

Algoritmo 10 Criba lineal

La ***entrada*** RFB, AFB, QCB, polinomio $f(x)$, raíz m de $f(x) \pmod N$, un entero C.
La ***respuesta*** lista de pares $rels = \{(a_0, b_0), \ldots, (a_t, b_t)\}$.

PASO 1: $b = 0$
PASO 2: $rels = [\,]$
PASO 3: Mientras $\sharp rels < \sharp RFB + \sharp AFB + \sharp QCB + 1$

a) $b = b + 1$

b) $a[i] = i + bm$ para $i \in [-C, C]$

c) para cada $(p, r) \in RFB$

- divida a la mayor potencia de p a partir de $a[j]$ donde $j = -bm + kp$ para $k \in \mathbb{Z}$ y $-C \leq j \leq C$

d) $e[i] = b^{\deg(f)} f(i/b)$ para $i \in [-C, C]$

e) para cada $(p, r) \in AFB$

- divida la mayor potencia de p a partir de $e[j]$ donde $j = -br + kp$ para $k \in \mathbb{Z}$ y $-C \leq j \leq C$

f) para $i \in [-C, C]$

- si $a[i] = e[i] = 1$ y $\text{mcd}(i, b) = 1$ agregar (i, b) a $rels$

g) $b = b + 1$

PASO 4: Salida $rels$

Acelerando la criba

Otro algoritmo de uso frecuente es la criba de retículas propuesta por John Pollard. Es similar al método propuesto por Morrison y Brillhart descrito en la Sección §6. Las bases de factores se dividen en pequeños subconjuntos y, entonces los elementos que son divisibles por un primo grande q son cribados. Esto acelera la criba a expensas de perder algunos elementos uniformes.

Una optimización que se puede hacer tanto para la criba lineal como para la de retículas es usar logaritmos para evitar divisiones. Esto significa que se debe guardar el logaritmo de las normas y restar $\ln(p)$ en lugar de dividirlo, debido a una de las leyes de logaritmos

$$\ln\left(\frac{N}{p}\right) = \ln(N) - \ln(p).$$

Por lo tanto, evita numerosas divisiones de números grandes, pero hay algunas cuestiones de las que se debe ser consciente en la optimización.

- No hay manera de detectar las potencias de los números primos p^j en las normas, por ejemplo si 3^7 es un factor en la norma sólo $\ln(3)$ se substraerá.
- Debido a las operaciones de punto flotante, los elementos correctos pueden ser anulados y los elementos malos pueden ser incluidos en el resultado.

El primer problema puede manejarse, en cierta medida, agregado una pequeña perturbación a los logaritmos. Si una norma no es RFB-uniforme, entonces debe tener un factor mayor que el primo más grande en RFB, así que si substraemos $\ln(\text{máx}(RFB))$ de todas las entradas, hay que elegir más elementos que son divisibles por potencias de primos sin la aprobación de los más equivocados.
Las dos cuestiones mencionadas anteriormente implican que los elementos encontrados deben ser divididos para asegurar que son RFB y AFB-uniformes.

§10.4. Álgebra lineal

De la criba obtenemos una lista de (a, b) que son RFB y AFB-uniformes, queremos encontrar un subconjunto para obtener un producto igual a un cuadrado.
Para que el producto sea cuadrado, los elementos tomados pueden tener incluso potencia impar en sus factores, pero al multiplicarlo con otro elemento que tiene el mismo factor y potencia impar forman un cuadrado, esta propiedad es la que usamos para encontrar dicho producto. Para aclarar consideremos como ejemplo la lista de números:

$$\{34, 89, 46, 32, 56, 8, 51, 43, 69\},$$

queremos encontrar un subconjunto de estos números tal que el producto sea cuadrado. Una solución es:

$$\{34, 46, 51, 69\},$$

ya que el producto es

$$34 \cdot 46 \cdot 51 \cdot 69 = 5503716 = 2^2 \cdot 3^2 \cdot 17^2 \cdot 23^2 = (2 \cdot 3 \cdot 17 \cdot 23)^2.$$

Esto es equivalente a resolver un sistema de ecuaciones lineales. El problema con esta operación son las dimensiones del sistema lineal, que puede resultar en una matriz con

dimensiones mayores a $10^9 \times 10^9$, y es difícil de representar en la memoria de una computadora moderna.

La matriz consiste en la factorización sobre los racionales, bases algebraicas y otra información derivada de la base de caracteres cuadrática con el signo de la norma. Dado que sólo estamos interesados en la paridad de las potencias de los elementos de la factorización, ponemos 1 si el elemento aparece como una potencia impar en la factorización y 0 si es par. Esto debería ser más claro en el Algoritmo 11 en el que se construye una matriz M sobre $\mathbb{F}_2$ con dimensiones $(\sharp relaciones) \times (\sharp RFB + \sharp AFB + \sharp QCB + 1)$.

Algoritmo 11 Construcción de la matriz

La ***entrada*** RFB, AFB, QCB, polinomio $f(x)$, raíz m de $f(x)$ (mód N), $rels = \{(a_0, b_0), \ldots, (a_t, b_t)\}$ de pares uniformes.
La ***respuesta*** matriz binaria M de dimensión $(\sharp rels) \times (\sharp RFB + \sharp AFB + \sharp QCB + 1)$.

PASO 1: Sea $M[i, j] = 0$
PASO 2: Para cada $(a_i, b_i) \in rels$

a) si $a_i + b_i m < 0$ hacer $M[i, 0] = 1$

b) Para cada $(p_k, r_k) \in$ RFB

- Sea l la potencia más grande de p_k que divide a $a_i + b_i m$
- Si l es par, hacer $M[i, 1 + k] = 1$

c) Para cada $(p_k, r_k) \in$ AFB

- Sea l la potencia más grande de p_k que divide a $(-b_i)^{\deg(f)} f(-\frac{a_i}{b_i})$
- Si l es par, hacer $M[i, 1 + \sharp\text{RFB } + k] = 1$

d) Para cada $(p_k, r_k) \in$ QCB

- Si el símbolo de Legendre $\left(\frac{a_i + b_i p_k}{r_k}\right) \neq 1$, hacer $M[i, 1 + \sharp\text{RFB } + \sharp\text{AFB } + k] = 1$

PASO 3: Salida M

La matriz es llevada a su forma escalonada para obtener la solución que producirá un cuadrado. El proceso para resolver el sistema lineal es utilizando el método de eliminación de Gauss. Es rápido y fácil de implementar pero tiene un inconveniente en este caso: como la matriz es muy grande, la memoria en uso puede ser insuficiente.

§10.5. Raíz cuadrada

Después del paso de álgebra lineal obtenemos una o más soluciones, es decir, los productos cuadrados pueden dar lugar a un factor trivial o no trivial de N.

Necesitamos determinar la raíz cuadrada de $S(m) \cdot f'(m)^2$ utilizando el Algoritmo (12) obteniendo el valor de Y el cual es nombrado **raíz cuadrada racional**, despues calcular el valor de X al utilizar el Algoritmo (13), el valor de X es nombrado **raíz cuadrada algebraica**.

El siguiente algoritmo muestra los pasos que se realizan para calcular la raíz cuadrada racional.

Algoritmo 12 Raíz cuadrada racional

La ***entrada*** N, polinomio $f(x)$, raíz m de $f(x)$, lista de elementos uniformes tales que su producto es un cuadrado $deps = \{(a_0, b_0), \ldots, (a_t, b_t)\}$.
La ***respuesta*** Entero Y.

PASO 1: Calcular $S(x) = \prod_{(a,b)\in deps}(a + b\,x)$ en $\mathbb{Z}[x]/\langle f(x)\rangle$
PASO 2: Respuesta $Y = \sqrt{S(m) \cdot f'(m)^2} \pmod N$

Encontrar un entero algebraico[†] está lejos de ser trivial, y es la parte más compleja del algoritmo GNFS pero no la que más tiempo consume, y la optimización de este paso no es tan importante. Existen algoritmos para factorizar polinomios en diversos campos, pero no muchos para factorizar en un campo de números.

Para esto se hace una aproximación y se utiliza una iteración de Newton para encontrar una raíz cuadrada.

El siguiente algoritmo es utilizado para calcular la raíz cuadrada algebraica.

[†] Un número $\alpha \in K$ es entero algebraico (sobre $\mathbb{Z}$) si es raiz de un polinomio mónico con coeficientes enteros

Algoritmo 13 Raíz cuadrada algebraica

La ***entrada*** N, polinomio $f(x)$, raíz m de $f(x)$, lista de elementos uniformes tales que su producto es un cuadrado $deps = \{(a_0, b_0), \ldots, (a_t, b_t)\}$.
La ***respuesta*** Entero X.

PASO 1: Calcular el producto $S(x)$ en $\mathbb{Z}[x]/\langle f(x)\rangle$ de los elementos de *deps*
PASO 2: Elegir un primo p grande (por ejemplo 2-5 veces más grande que N)
PASO 3: Elegir $r(x) \in \mathbb{Z}_p[x]/\langle f(x)\rangle$ aleatorio con $\deg(r) = \deg(f) - 1$
PASO 4: Calcular $R_0 + R_1 y = (r(x) - y)^{\frac{p^d-1}{2}} \in (\mathbb{Z}_p[x]/\langle f(x)\rangle)[y]/(y^2 - S(x))$, es decir calcular la potencia $\frac{p^d-1}{2}$ de $r(x)$ módulo $y^2 - S(x)$
PASO 5: Si $S(x)R_1^2 \neq 1$ ir al paso 2 y elegir otro p o $r(x)$
PASO 6: Sea $k = 0$
PASO 7: Sea $k = k + 1$
PASO 8: Calcular $R_{2k} = \frac{R_k(3 - S(x)R_k^2)}{2} \pmod{p^{2k}}$
PASO 9: Si $(R_k S(x))^2 \neq S(x)$ ir al paso 7
PASO 10: Calcular $s(x) = \pm S(x)R_k$
PASO 11: Respuesta $X = s(m) \cdot f'(m) \pmod{N}$

Por ultimo una vez calculado la raíz cuadrada racional Y y la raíz cuadrada algebraica X, se tiene que calcular $\mathrm{mcd}(N, X - Y)$ y $\mathrm{mcd}(N, X + Y)$ para encontrar los factores, recordemos que el calculo de $\mathrm{mcd}(N, X - Y)$ y $\mathrm{mcd}(N, X + Y)$ son factores triviales o no triviales de N.

§11. Factorización con formas cuadráticas

Un polinomio $ax^2 + bxy + cy^2 \in \mathbb{Z}[x, y]$ es llamado una **forma cuadrática binaria** y $\triangle = b^2 - 4ac$ su **discriminante**. La forma cuadrática binaria es denotada por (a, b, c). Si $\mathrm{mcd}(a, b, c) = 1$ es nombrada como **primitiva**, si $a > 0$ y $\triangle < 0$ es llamada **positiva**. Dos formas (a, b, c) y (a', b', c') son **equivalentes** si existe $\alpha, \beta, \gamma, \delta \in \mathbb{Z}$ con $\alpha\delta - \beta\gamma = 1$ tal que $a'u^2 + b'uv + c'v^2 = ax^2 + bxy + cy^2$, donde $u = \alpha x + \gamma y$, $v = \beta x + \delta y$. Escribimos formas equivalentes como $(a, b, c) \sim (a', b', c')$. Las formas equivalentes tienen el mismo discriminante.
Ahora $\triangle \in \mathbb{Z}_{\leq -1}$, $\triangle \equiv 0 \pmod 4$ o $\triangle \equiv 1 \pmod 4$. Si consideramos la forma de discriminante $\triangle$, entonces $c = (b^2 - \triangle)/(4a)$ con $a \neq 0$, por lo que es suficiente con escribir (a, b) para denotar la forma cuadrática (a, b, c). Se estudia el conjunto $C_\triangle$ de clases de equivalencia de formas cuadráticas. En primer lugar desarrollaremos una forma conveniente de representar los elementos de $C_\triangle$: La forma (a, b, c) es llamada **reducida** si $|b| \leq a \leq c$, y $b \geq 0$ si $b = a$ o $a = c$.

Proposición II.1. Cada clase en $C_\triangle$ contiene precisamente una forma reducida.

Prueba: El siguiente algoritmo da la pauta para probar la proposición anterior.

Algoritmo 14 La reducción de la forma cuadrática binaria positiva

La ***entrada*** La forma (a, b).
La ***respuesta*** La forma reducida (a, b).

PASO 1: Reemplazar (a, b) por $(a, b - 2k \cdot a)$ donde $k \in \mathbb{Z}$ es tal que $-a < b - 2k \cdot a \leq a$
PASO 2: Si (a, b, c) es reducido, entonces parar; en otro caso reemplazar (a, b, c) por $(c, -b, a)$ e ir al paso 1.

Después del paso 1 tenemos $|b| \leq a$ y $b \geq 0$ si $|b| = a$. Si (a, b, c) no es reducida después de este paso entonces $a > c$ o $b < 0$ y $a = c$. En este último caso (a, b, c) será reducida después del paso 2. En el primer caso a disminuye por lo menos en 1, lo que muestra que es finito el algoritmo. □

Se define una estructura de grupo sobre $C_\triangle$: identificamos la clase con su representante reducido. Para calcular $(a_1, b_1) \cdot (a_2, b_2)$ usamos el algoritmo extendido de Euclides para determinar

$$d = \text{mcd}\left(a_1, a_2, \frac{b_1 + b_2}{2}\right) \tag{2.3}$$

y $r, s, t \in \mathbb{Z}$ tal que

$$d = r \cdot a_1 + s \cdot a_2 + t \cdot \frac{b_1 + b_2}{2}. \tag{2.4}$$

El producto obtenido es

$$\left(\frac{a_1 a_2}{d^2}, b_2 + \frac{2a_2(s\frac{b_1 - b_2}{2} - tc_2)}{d}\right) \tag{2.5}$$

donde $c_2 = (b_2^2 - \triangle)/(4a_2)$. El elemento identidad es

$$1_\triangle = \begin{cases} (1, 1), & \text{si } \triangle \text{ es impar} \\ (1, 0), & \text{si } \triangle \text{ es par.} \end{cases}$$

El inverso de (a, b) es $(a, -b)$.

Proposición II.2. $C_\triangle$ es un grupo abeliano finito.

Prueba: $(a_1, b_1) \cdot (a_2, b_2)$ es lo que se tiene en (2.5). Luego de (2.4) se despeja $s \cdot a_2$, se sustituye $s \cdot a_2$ y c_2 en (2.5), simplificando los terminos resulta:

$$\left(\frac{a_1a_2}{d^2}, b_1 + \frac{2a_1(r\frac{b_2-b_1}{2} - tc_1)}{d}\right) = (a_2, b_2)\cdot(a_1, b_1)$$

No es facil mostrar que $C_\triangle$ es finito. □

Una forma reducida se denomina **ambigua** si $b = 0$ o $a = b$ o $a = c$. Si conocemos la forma ambigua podemos obtener la factorización de $\triangle$

si $(a, b) = (a, 0)$	entonces $\triangle = b^2 - 4ac = -4ac$
si $(a, b) = (a, a)$	entonces $\triangle = a^2 - 4ac = a(a - 4c)$
si $(a, b, c) = (a, b, a)$	entonces $\triangle = b^2 - 4a^2 = (b - 2a)(b + 2a)$

Nótese que debido a la forma cuadrática los factores deben ser coprimos. En consecuencia, sea $\triangle \equiv 1 \pmod 4$ y $|\triangle| = p\cdot q$ una factorización en los factores coprimos. Se desea obtener una forma ambigua, lo que corresponde a esta factorización. Para ello se tiene que resolver cualquiera de las ecuaciones.

$$-p\cdot q = \triangle = a(a - 4c) \tag{2.6}$$

o

$$-p\cdot q = \triangle = (b - 2a)(b + 2a) \tag{2.7}$$

El primer tipo de forma ambigua es inútil en este contexto porque el discriminante correspondiente es par.
Ya que $-p\cdot q \equiv 1 \pmod 4$ debemos tener $-p \equiv 3 \pmod 4$ y $q \equiv 3 \pmod 4$ o $-p \equiv 1 \pmod 4$ y $q \equiv 1 \pmod 4$, en cualquier caso $p + q \equiv 0 \pmod 4$. Para satisfacer la Ecuación (2.6) podemos hacer $a = p = b,\ c = (p^2 + p\cdot q)/4p = (p + q)/4$.
Cuando la forma es reducida, satisface $|b| \le a \le c$, es decir $q \le p \le (p + q)/4$. La primera desigualdad se satisface trivialmente. Para que el orden de la segunda desigualdad se mantenga, necesitamos $3p \le q$. Finalmente, ya que $|b| = a$ debemos tener $b \ge 0$, que también es cierto.
Ahora examinemos las formas ambiguas del tipo 3: escogemos $a = (p+q)/4,\ b = (q-p)/2$. De nuevo examinemos cuándo esta forma cuadrática es reducida. En $|b| \le a \le c$ la segunda desigualdad es trivial. La primera es $(p - q/2) \le (p + q)/4$ (asumimos sin pérdida de generalidad que $q < p$). Por lo tanto $2q - 2p \le p + q$, $q \le 3p$. Esto cubre el resto de los casos, se tiene así el siguiente resultado.

Proposición II.3. Para $\triangle < 0$, $\triangle \equiv 1 \pmod 4$ hay una correspondencia 1-1 entre las formas ambiguas y la factorización de $\triangle$ en factores coprimos: si $-\triangle = p\,q$, $p < q$, entonces la correspondencia está dada por

$$p\,q \leftrightarrow \begin{cases} (p,p), & \text{para } q > 3p \\ (\frac{p+q}{4}, \frac{q-p}{2}), & \text{para } q < 3p \end{cases}$$

Ver la prueba en [4].

Proposición II.4. Una clase en $C_\triangle$ contiene formas ambiguas de orden 1 o 2 en $C_\triangle$.

Prueba: Considerando la forma (a,a). Aplicando el algoritmo de composición:
$d = \text{mcd}(a,a,a) = a$, $r = 1$, $s = t = 0$ resulta $(1,0) \in C_\triangle$. □

Por lo tanto la factorización de $\triangle$ se puede obtener de los elementos de orden 2 en $C_\triangle$. Un conjunto de formas primitivas G es llamado **conjunto generador** para $C_\triangle$ si cada elemento en $C_\triangle$ es producto de potencias de las clases de las formas en G. Los elementos de orden ≤ 2 forman un subgrupo $C_{2,\triangle}$ de $C_\triangle$. Si $\{f_1, \ldots, f_r\}$ es un conjunto de generadores para el grupo $C_\triangle$, entonces el conjunto generador para el subgrupo de todos los elementos de orden par se puede obtener de la siguiente forma: determinar un divisor $\widetilde{h}$ del número de clases $h_\triangle = |C_\triangle|$. Sea h' la parte impar de $\widetilde{h}$. Calcular $f_i' = f_i^{h'}$ para $1 \leq i \leq r$. Entonces f_i' genera el subgrupo de todos los elementos de orden par en $C_\triangle$.
Nota: Si $h_\triangle$ es impar, lo anterior no funciona porque $C_\triangle$ es abeliano y no tendrá elementos de orden par.
Ahora determinar para $1 \leq i \leq r$ (si existe) un exponente $m_i \in \mathbb{Z}_{\geq 1}$ tal que

$$\begin{aligned} f_i'^{2^{m_i}} &= 1 \\ f_i'^{2^{m_i-1}} &\neq 1 \end{aligned}$$

(en otro caso sea $m_i = 0$) y sea $\widetilde{f}_i = f_i'^{2^{m_i-1}}$. Entonces $\{\widetilde{f}_1, \ldots, \widetilde{f}_r\}$ es un conjunto generador para $C_{2,\triangle}$. Por lo tanto

$$C_{2,\triangle} = \{\widetilde{f}_1^{x_1} \ldots \widetilde{f}_r^{x_r} | x_i \in \{0,1\}\}.$$

Queda por determinar un método para calcular un divisor de $h_\triangle = |C_\triangle|$. Con esta propuesta volvemos a hacer un conjunto generador $\{f_1, \ldots, f_r\}$ para el grupo de clases $C_\triangle$.

Proposición II.5. **(i)** El mapeo

$$\begin{aligned} \mathbb{Z}^r &\to C_\triangle \\ (x_1, \ldots, x_r) &\to \prod_{i=1}^{r} f_i^{x_i} \end{aligned}$$

es un epimorfismo.

(ii) El núcleo de este mapeo es una retícula r-dimensional en $\mathbb{Z}^r$ de determinante $h_\triangle$.

La prueba se presentara más adelante.

§11.1. Retículas enteras

Una **retícula entera** es un subgrupo (abeliano aditivo) del grupo $\mathbb{Z}^r$ para algún $r \in \mathbb{Z}_{\geq 1}$.

Proposición II.6. Sea $L \subseteq \mathbb{Z}^r$ una retícula entera. Entonces L es de la forma

$$\begin{aligned} L &= \mathbb{Z}\underline{v}_1 \oplus \ldots \oplus \mathbb{Z}\underline{v}_k \\ &= \left\{ \sum_{j=1}^{k} x_j \underline{v}_j | x_j \in \mathbb{Z}, 1 \leq j \leq k \right\} \end{aligned}$$

donde $\underline{v}_1, \ldots, \underline{v}_k \in \mathbb{Z}^r$ son linealmente independientes. El conjunto ordenado $\{\underline{v}_1, \ldots, \underline{v}_k\}$ es llamado **base** de L.

Prueba: Para $1 \leq j \leq r$ sea $\underline{v}_j$ un vector en L de la forma $\underline{v}_j = (v_1^{(j)}, \ldots, v_j^{(j)}, 0, \ldots, 0)$ con un mínimo $v_j^{(j)} > 0$. Si tal vector no existe, entonces sea $\underline{v}_j = 0$. Pedimos que el vector no-cero $\underline{v}_j$ forme parte de la base de L. De hecho, sea $\underline{v} \in L$. Entonces podemos encontrar por inducción coeficientes $\lambda_j \in \mathbb{Z}$ tales que para $1 \leq j_0 \leq r$ tenemos

$$\underline{v} - \sum_{j=1}^{r} \lambda_j \underline{v}_j = \underline{w}_{j_0} = (w_1^{(j_0)}, \ldots, w_{j_0}^{(j_0)}, 0, \ldots, 0)$$

y esto demuestra la afirmación. Así que hemos encontrado $\lambda_{j_0+1}, \ldots, \lambda_r$ para $j_0 \leq r$. Si $v_{j_0}^{(j_0)} = 0$, entonces también tenemos $w_{j_0}^{(j_0)} = 0$. Si $v_{j_0}^{(j_0)} \neq 0$, se divide con el residuo: $w_{j_0}^{(j_0)} = \lambda_{j_0} v_{j_0}^{(j_0)} + z$ con $0 \leq z < v_{j_0}^{(j_0)}$. Aquí $z = 0$ debido a la minimalidad de $v_{j_0}^{(j_0)}$. □

Proposición II.7. Sea $L \subseteq \mathbb{Z}^r$ una retícula entera. Entonces todas las bases de L tienen la misma cardinalidad k, la cual es llamada la **dimensión** de L.

Prueba: Sea $\{u_1, u_2, \ldots, u_k\}$, y $\{v_1, v_2, \ldots, v_l\}$ bases de L. Cualquier subconjunto linealmente independiente de $\langle u_1, u_2, \ldots, u_k \rangle$ a lo más puede tener k elementos, es decir, $l \leq k$. Lo mismo pasa con $\{v_1, v_2, \ldots, v_l\}$, cualquier subconjunto linealmente independiente de $\langle v_1, v_2, \ldots, v_k \rangle$ a lo más puede tener l elementos, es decir, $k \leq l$. Por lo tanto $k = l$. □

Sea $\{\underline{v}_1, \ldots, \underline{v}_k\}$ una base de L y sean $\underline{w}_1, \ldots, \underline{w}_k \in L$ tales que

$$\underline{w}_j = \sum_{i=1}^{k} \lambda_{ij} \underline{v}_i$$

con $(\lambda_{ij}) \in \mathbb{Z}^{k\times k}$. Entonces $\{\underline{w}_1, \ldots, \underline{w}_k\}$ es una base de L si y sólo si $(\lambda_{ij}) \in GL(n, \mathbb{Z})$, es decir $|\det((\lambda_{ij}))| = 1$.
Sea $L \subseteq \mathbb{Z}^r$ una retícula entera. Los vectores $\underline{v}_1, \underline{v}_2 \in \mathbb{Z}^r$ son llamados **congruentes módulo** L si $\underline{v}_1 - \underline{v}_2 \in L$. Esto es una relación de equivalencia en $\mathbb{Z}^r$. Las clases de equivalencia de $\underline{v} \in \mathbb{Z}^r$ son $\underline{v} + L$. El conjunto $\{\underline{v} + L | \underline{v} \in \mathbb{Z}^r\}$ forma un grupo con respecto a la adición:

$$(\underline{v} + L) + (\underline{v}' + L) = (\underline{v} + \underline{v}') + L$$

Este grupo es el grupo factor $\mathbb{Z}^r/L$. Un conjunto $V \subseteq L$ tal que $\{\underline{v} + L | \underline{v} \in V\} = \mathbb{Z}^r/L$ es llamado un **conjunto de representantes** de $\mathbb{Z}^r/L$. Si para cada $\underline{v}_0 \in V$ el conjunto $V \backslash \{\underline{v}_0\}$ no es un conjunto de representantes, entonces V es llamado **conjunto minimal de representantes** de $\mathbb{Z}^r/L$.
El **determinante de una retícula entera** k-dimensional $L \subseteq \mathbb{Z}^r$ es definido como

$$d(L) = (|\det((\underline{v}_i, \underline{v}_j))_{1\leq i,j\leq k}|)^{\frac{1}{2}},$$

donde $\{\underline{v}_1, \ldots, \underline{v}_k\}$ es cualquier base de L y $(\underline{v}_i, \underline{v}_j)$ denota el producto interno de $\underline{v}_i$ y $\underline{v}_j$:

$$(\underline{v}_i, \underline{v}_j) = \sum_{l=1}^{k} (\underline{v}_{l\,i} \underline{v}_{l\,j})$$

Proposición II.8. Si $L \subseteq \mathbb{Z}^r$ es una retícula entera r-dimensional, entonces

$$d(L) = |\det(\underline{v}_1, \ldots, \underline{v}_r)|$$

para cualquier base $\{\underline{v}_1, \ldots, \underline{v}_r\}$ de L.

Prueba: si $\{v_1, \ldots, v_r\}$ y $\{u_1, \ldots, u_r\}$ son bases de L, entonces existe $(\lambda_{ij}) \in GL(r, \mathbb{Z})$ tal que

$$((v_i, v_j)) = (\lambda_{ij})^{-1}((u_i, u_j))(\lambda_{ij}).$$

Por lo tanto

$$det((v_i, v_j)) = det(\lambda_{ij})^{-1}\, det((u_i, u_j))\, det(\lambda_{ij}) = det(u_i, u_j).$$

□

Proposición II.9. Sea $L \subseteq \mathbb{Z}^r$ una retícula entera. Entonces $\mathbb{Z}^r/L$ es finito si y sólo si la dimensión de L es r. En este caso tenemos $d(L) = |\mathbb{Z}^r/L|$

Prueba: Se define $\underline{v}_1, \ldots, \underline{v}_r$ como en la prueba de la Proposición (II.6). Entonces

$$\left\{ \sum_{i=1}^{r} x_i \underline{v}_i | 0 \leq x_i < v_i^{(i)} \text{ para } v_i^{(i)} \neq 0 \text{ y } x_i \in \mathbb{Z} \text{ para } v_i^{(i)} = 0 \right\}$$

es un conjunto minimal de representantes para $\mathbb{Z}^r/L$. Lo que confirma la afirmación. □

Prueba de la Proposición (II.5)

(i) El mapeo es claramente un homomorfismo. Es sobreyectiva porque $\{f_1, \ldots, f_r\}$ es un conjunto generador.

(ii) Sea L el núcleo del mapeo definido en el enunciado de la Proposición (II.5) (ii). Entonces L es un subgrupo de $\mathbb{Z}^r$, por lo tanto una retícula entera. Por el Primer Teorema de Homomorfismos tenemos $\mathbb{Z}^r/L \cong C_\triangle$. Con $C_\triangle$ finito, el grupo factor $\mathbb{Z}^r/L$ también es finito y por la Proposición (II.9) L es de dimensión r. Por otra parte tenemos por la Proposición (II.9): $|C_\triangle| = h_\triangle = |\mathbb{Z}^r/L| = d(L)$. □

Si $L' \subset L \subset \mathbb{Z}^r$ son retículas enteras, entonces L' es llamada **subretícula** de L. Además, podemos definir el grupo factor L/L'.

Proposición II.10. Si $L' \subseteq \mathbb{Z}^r$ es una subretícula de la retícula entera $L \subseteq \mathbb{Z}^r$ de la misma dimensión que L, entonces $d(L') = c \cdot d(L)$ para un $c \in \mathbb{Z}_{\geq 1}$.

Prueba: Sea $\{\underline{v}_1, \ldots, \underline{v}_r\}$ una base de L donde $\underline{v}_i = (v_{1i}, v_{2i}, \ldots, v_{ri})^t$ y $\{\underline{w}_1, \ldots, \underline{w}_r\}$ una base de L' donde $\underline{w}_j = (w_{1j}, w_{2j}, \ldots, w_{rj})^t$. Como $L' \subseteq L$ se puede escribir cada $\underline{w}_j$ como combinacion lineal de $\underline{v}_i$, con $i, j = 1, 2, \ldots, r$, entonces

$$\begin{aligned} (w_{1j}, w_{2j}, \ldots, w_{rj}) &= a_{1j}(v_{11}, v_{21}, \ldots, v_{r1}) \\ &+ a_{2j}(v_{12}, v_{22}, \ldots, v_{r2}) \\ &\vdots \\ &+ a_{rj}(v_{1r}, v_{2r}, \ldots, v_{rr}) \end{aligned}$$

así

$$\begin{aligned} w_{1j} &= a_{1j}v_{11} + a_{2j}v_{12} + \ldots + a_{rj}v_{1r} \\ w_{2j} &= a_{1j}v_{21} + a_{2j}v_{22} + \ldots + a_{rj}v_{2r} \\ &\vdots \\ w_{rj} &= a_{1j}v_{r1} + a_{2j}v_{r2} + \ldots + a_{rj}v_{rr} \end{aligned}$$

lo anterior lo podemos ver como

$$\begin{pmatrix} w_{1j} \\ w_{2j} \\ \vdots \\ w_{rj} \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1r} \\ v_{21} & v_{22} & \dots & v_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ v_{r1} & v_{r2} & \dots & v_{rr} \end{pmatrix} \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{rj} \end{pmatrix}$$

simplificando la notación obtenemos

$$\underline{w}_j = \begin{pmatrix} \underline{v}_1 & \underline{v}_2 & \dots & \underline{v}_r \end{pmatrix} \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{rj} \end{pmatrix}$$

entonces

$$\begin{pmatrix} \underline{w}_1 & \underline{w}_2 & \dots & \underline{w}_r \end{pmatrix} = \begin{pmatrix} \underline{v}_1 & \underline{v}_2 & \dots & \underline{v}_r \end{pmatrix} A$$

donde $A = (a_{ij})$, por lo tanto

$$\begin{aligned} d(L') &= |det(\underline{w}_1, \underline{w}_2, \dots, \underline{w}_r)| \\ &= |det(\underline{v}_1, \underline{v}_2, \dots, \underline{v}_r)\, A| \\ &= |det(\underline{v}_1, \underline{v}_2, \dots, \underline{v}_r)||det(A)| \\ &= d(L)\, c \end{aligned}$$

con $c = |det(A)|$, como los valores de a_{ij} son enteros con $1 \leq i, j \leq r$, entonces $c \in \mathbb{Z}$, como $det(A) \neq 0$ y por el valor absoluto entonces $c > 0$. □

El número de clases $h_\triangle$ es el determinante de L, con L una retícula entera de dimensión r. Si encontramos r vectores linealmente independientes $\underline{w}_1, \dots, \underline{w}_r \in L$, entonces $L' = \oplus_{i=1}^{r} \mathbb{Z}\underline{w}_i$ es una subretícula r-dimensional de L, es decir por la Proposición (II.10) el determinante $d(L')$ está dado por un divisor de $d(L) = h_\triangle$. Por lo tanto para encontrar el orden de un divisor $h_\triangle$ es suficiente con determinar r vectores linealmente independientes en L. Estos vectores serán generados por un algoritmo probabilístico.

§11.2. Formas principales

Para un número primo p con $(\frac{\triangle}{p}) = 1$ (simbolo de Legendre [2]) hay un $b_p \in \mathbb{Z}_{\leq 1}$ tal que (p, b_p) es una forma de discriminante $\triangle$. Para hacer b_p único escogemos $b_p = \text{mín}\{b \in \mathbb{Z}_{>0} | b^2 \equiv \triangle \pmod{4p}\}$. La forma reducida I_p equivalente a (p, b_p) es llamada la **forma principal**.

Proposición II.11. Las formas principales I_p con $p \leq 2(\ln|\triangle|)^2$ generan el grupo de clases $C_\triangle$.

Ver la prueba en [12].

Nótese que la última proposición asegura la existencia de un conjunto de generadores reducidos para $C_\triangle$.

Proposición II.12. Sea (a,b) una forma de discriminante $\triangle$ y $\mathrm{mcd}(\triangle, a) = 1$. Sea $a = \prod_{p\in\mathbb{P}} p^{e_p}$ la factorización de a donde $\mathbb{P}$ es el conjunto de primos. Tenemos que si $e_p \neq 0$ entonces $(\frac{\triangle}{p}) = 1$ y $(a,b) \sim \prod_{p\in\mathbb{P}} I_p^{s_p e_p}$ donde $s_p \in \{-1,1\}$ es determinado por $b \equiv s_p \cdot b_p \pmod{2p}$.

Prueba: Como $\triangle = b^2 - 4ac$ y $a = \prod_{p\in\mathbb{P}} p^{e_p}$ con $e_p \neq 0$ se tiene que $-4ac$ es divisible por p entonces $\triangle \equiv b^2 \pmod{p}$ para todo $p \in \mathbb{P}$, por lo tanto $(\frac{\triangle}{p}) = 1$.

Para probar lo segundo en primer lugar observamos que $I_p^{s_p}$ es equivalente a la forma (p, b') con $b' \equiv b \pmod{2p}$. También $I_p^{s_p}$ es equivalente a la forma (p^{e_p}, b'') con $b'' \equiv b \pmod{2p^{e_p}}$. Ya que $\mathrm{mcd}(\triangle, a) = 1$ se sigue $p \not| b_p = (b_p + b_p)/2$. Por lo tanto, al utilizar el Algoritmo (14) produce una forma equivalente a (p^{e_p}, b') donde $b'^2 \equiv \triangle \pmod{4p^{e_p}}$. Para p impar el grupo $(\mathbb{Z}/2p^{e_p}\mathbb{Z})^*$ es cíclico. Por lo tanto la congruencia $x^2 \equiv \triangle \pmod{2p^{e_p}}$ tiene como máximo dos soluciones. Puesto que b es una solución y $b \equiv b'' \pmod{2p}$, se deduce que $b \equiv b'' \pmod{2p^{e_p}}$. Un argumento similar demuestra la afirmación para $p = 2$. □

La idea de producir las relaciones aleatorias en el grupo de clases es el siguiente. Fijamos $C_1, C_2 \in \mathbb{Z}_{>1}$ y sea $p_1, \ldots, p_r$ la sucesión de números primos menores que C_1 con $(\frac{\triangle}{p_i}) = 1$ ordenados de acuerdo a su tamaño.

Elija $\underline{y} = (y_1, \ldots, y_r) \in \mathbb{Z}^r$ al azar y calcule la forma reducida como $f = f(\underline{y}) = (a(\underline{y}), b(\underline{y}))$ en la clase de $\prod_{i=1}^r I_{p_i}^{y_i}$. Trate de factorizar $a = a(\underline{y})$ sobre la base de factor $F = \{p_1, \ldots, p_r\}$. Supongamos que esta factorización es exitosa:

$$a(\underline{y}) = \prod_{i=1}^{r} p_i^{e_i(\underline{y})}.$$

Determine $s_i(\underline{y}) \in \{\pm 1\}$ de acuerdo a la condición de congruencia

$$b(\underline{y}) \equiv s_i(\underline{y}) b_{p_i} \pmod{2p_i} \quad 1 \leq i \leq r,$$

Entonces sabemos de la Proposición (II.12) que $(a(\underline{y}), b(\underline{y}))$ está en la clase de

$$\prod_{i=1}^{r} I_{p_i}^{s_i(\underline{y}) e_i(\underline{y})}$$

y por la construcción, esta forma queda en la misma clase que

$$\prod_{i=1}^{r} I_{p_i}^{y_i}.$$

Por lo tanto $\underline{x} = (x_1, \ldots, x_r)$ con

$$x_i = y_i - s_i(\underline{y})e_i(\underline{y}) \ \ (1 \leq i \leq r)$$

pertenece al núcleo del mapeo.
Después de haber obtenido una cantidad suficiente de relaciones aleatorias encontraremos r vectores linealmente independientes en el núcleo y por lo tanto un múltiplo del número de clases, lo que nos permite factorizar $\triangle$.
Aunque este algoritmo produce la factorización de N, tiene un gran inconveniente: el tiempo de ejecución puede ser muy grande al calcular el determinante de la relación de vectores. Sin embargo, existe una modificación, que evita este inconveniente:
Calcular $r+1$ vectores $\underline{x}_1, \ldots, \underline{x}_{r+1}$. Entonces debe haber coeficientes $c_i \in \{0, 1\}$ $(1 \leq i \leq r+1)$ tal que

$$\sum_{i=1}^{r+1} c_i \underline{x}_i \equiv \underline{0} \pmod 2.$$

Los coeficientes se pueden determinar mediante el cálculo usual sobre $\mathbb{F}_2$. Una vez que conocemos todos los c_i, se determinan los elementos del núcleo

$$\underline{x} = \sum_{i=1}^{r+1} c_i \underline{x}_i$$

y se escribe

$$\underline{x} = 2\underline{x}' = 2(x_1', \ldots, x_r').$$

A continuación, la forma reducida en la clase de $\prod_{i=1}^{r} f_i x_i'$ tiene ya sea orden 1 o 2 en la clase del grupo. Esto significa que se ha encontrado una forma ambigua y por lo tanto un factor de N si tenemos suerte. Si no, repetimos el procedimiento.

§12. Algoritmo de Shor

El algoritmo de Peter Shor fue formulado en 1994, está diseñado para tomar ventaja del potencial inherente de la computadora cuántica (en contraste con la clásica), explota un método de factorización que difiere de las cribas que en la actualidad se emplean para factorizar números grandes.
Sea n coprimo con $N = pq$, donde p y q son dos números primos distintos lo suficientemente grandes. Sea f_j, $j = 1, 2, 3, ..,$ el residuo cuando n^j es dividido por N.

$$n^j \equiv f_j \pmod{pq} \tag{2.8}$$

con $0 < f_j < N$. La congruencia

$$n^{\phi(N)} = n^{(p-1)(q-1)} \equiv 1 \pmod{pq}, \tag{2.9}$$

implica $f_\phi = 1$[†]. Sin embargo, para n dado, pueden existir otros enteros $1 \leq j \leq \phi(N) = (p-1)(q-1)$ para los cuales $f_j = 1$. El más pequeño de tales j, será denotado por r, es llamado el **orden de** n **módulo** pq. Por lo tanto la Congruencia (2.8) para $j = r$ es reescrita como

$$n^r - 1 \equiv 0 \pmod{pq}. \tag{2.10}$$

Supongamos que el orden r de algún entero $n < N$ coprimo con N es conocido. Supongamos además que r es par, necesario para que $r/2$ sea entero y así tenga sentido emplear congruencias. La Congruencia (2.10) implica

$$(n^{r/2} - 1)(n^{r/2} + 1) \equiv 0 \pmod{pq}. \tag{2.11}$$

Por definición r es la potencia más pequeña de n para la cual se cumple la Congruencia (2.10). Entonces el factor $(n^{r/2} - 1)$, en el lado izquierdo de la Congruencia (2.11), no puede ser dividido exactamente por pq, es decir,

$$n^{r/2} - 1 \not\equiv 0 \pmod{pq}. \tag{2.12}$$

El segundo factor de la Congruencia (2.11), no está sujeto a tal restricción, es decir, es posible que

$$n^{r/2} + 1 \equiv 0 \pmod{pq}. \tag{2.13}$$

No es necesario que la Congruencia (2.13) se cumpla, por lo cual suponemos

$$n^{r/2} + 1 \not\equiv 0 \pmod{pq}. \tag{2.14}$$

Si ambas Congruencias (2.12) y (2.14) son ciertas, tenemos que el producto en el lado izquierdo de la Congruencia (2.11) es precisamente divisible por pq, aunque ningún factor en este producto es divisible por pq.

Para evitar la contradicción, uno de los factores en el producto del lado izquierdo de la Congruencia (2.11), digamos el factor $(n^{r/2} - 1)$, debe ser divisible por p pero no por q, mientras el otro factor $(n^{r/2} + 1)$, es divisible por q pero no por p. Cuando el orden r de n módulo N es par, las Congruencias (2.12) y (2.14) implican que la clave dada $N = pq$ es inmediatamente factorizada al calcular el máximo común divisor de N y cada uno de los factores de $n^r - 1$: $\text{mcd}(N, n^{r/2} + 1) = q$ y $\text{mcd}(N, n^{r/2} - 1) = p$. Alternativamente, se puede factorizar N calculando primero q, de la forma antes mencionada y después calcular p mediante la división de N entre q.

[†] La Congruencia (2.9) garantiza que el conjunto $\{k \mid n^k \equiv 1 \pmod{pq}\}$ es distinto del conjunto vacío.

Calcular el máximo común divisor de una pareja de números grandes requiere de un tiempo factible. Por consiguiente, factorizar un número grande $N = pq$ depende principalmente de determinar el orden r de n módulo N.
Si se contara con una computadora cuántica con la cual se pudiera determinar r, factorizar N se volvería factible al usar la propiedad de periodicidad de la sucesión $f_j,\ j = 1, 2, 3, \ldots$, definida por la Congruencia (2.8). Para cualquier n, todos los enteros $f_1, f_2, \ldots, f_{r-1}, f_r = 1$ son diferentes para j en el rango $1 \leq j \leq r$, y para cada entero positivo k, se tiene $f_j = f_{j+r} = f_{j+2r} = \cdots = f_{j+kr} = f_{j+(k+1)r} = \cdots$. En otros términos, la sucesión $f_j,\ j = 1, 2, 3, \ldots$, es periódica con periodo r.
Por ejemplo para $N = 55$ y $n = 16$ el orden es $r = 5$ y la sucesión f_j es (empezando con $j = 1$) $16, 36, 26, 31, 1, 16, 36, 26, 31, 1, 16, 36, \ldots$, similarmente para $n = 12$, el orden es $r = 4$ y la sucesión f_j es $12, 34, 23, 1, 12, 34, 23, 1, 12, 34, \ldots$.
Lo conveniente es utilizar el algoritmo de Shor para conocer el valor de r, y así factorizar N. Refiriéndose a nuestro ejemplo, escogiendo $n = 12$ y utilizando el algoritmo de Shor resulta $r = 4$. Se obtiene $f_{r/2} = f_2 = 34$ (es congruente a 12^2 módulo 55) para que no haya contradicción en la Congruencia (2.11) las Congruencias (2.12) y (2.14) satisfacen para f_2, verificamos que $f_2 + 1 = 35$ debe ser divisible por uno de los factores de 55 (en este caso 5, cuando determináramos el valor de $\text{mcd}(35, 55)$ y que $f_2 - 1 = 33$ debe ser divisible por el otro factor de 55 (en este caso 11), cuando se determina el valor de $\text{mcd}(33, 55)$ o (más simplemente) por la división directa de 55 por su factor ya determinado.

§12.1. Operaciones que constituyen el algoritmo de Shor

El algoritmo de Shor busca un periodo r en la sucesión f_j, $j = 1, 2, 3, \ldots$, obtenida de la Congruencia (2.8) dado n coprimo con N. En la práctica el orden r puede lograr su posible valor máximo en $(p-1)(q-1)/2$, para N grande es probable que pueda ser ligeramente más pequeño que la mitad de $N = pq$. Para nuestro ejemplo $N = 55$, el orden de r alcanza su posible valor máximo en $((p-1)(q-1)/2 = 20)$, al elegir un número $n < 55$ coprimo con 55, el número n puede ser tan pequeño como 2 y tan grande como 53. Por consiguiente, determinar r al usar el algoritmo de Shor requiere del uso de potencias como se muestra en la Congruencia (2.8). Shor recomienda que la máxima potencia empleada j sea por lo menos $j_{\text{máx}} = N^2$. Williams y Clearwater [15] recomiendan que $j_{\text{máx}}$ sea aún mayor, al menos $2N^2$.

Determinar el tamaño mínimo de qbits requerido por la computadora: división de los qbits en dos registros.
La computadora cuántica a emplear para determinar r a través del algoritmo de Shor debe contener suficientes qbits por lo menos para representar a $j_{\text{máx}} = N^2$. El número mínimo de qbits necesario para representar a N^2 es $\lfloor \log_2 N^2 \rfloor + 1$. Así, la computadora cuántica debe contener un conjunto de $y = \lfloor \log_2 N^2 \rfloor + 1$ qbits, que será el registro Y. Además, la computadora debe contener un segundo juego de qbits, donde se registrará a Z, capaz de acumular los valores calculados de f_j, que pueden ser tan grandes como $N - 1$. El

tamaño de este registro será de $z = \lfloor \log_2(N-1) \rfloor + 1$ qbits. Notemos que N se conoce por ser el producto de un par de primos diferentes y así no es una potencia de 2, por lo que $2^{y-1} < N^2 < 2^y$.
Para N grande tanto la diferencia entre $\log_2(N^2) + 1$ y $2L = \log_2(N^2)$, como la diferencia entre $\lfloor \log_2(N-1) \rfloor$ y L o entre L y $L+1$, es despreciable con el propósito de estimar el esfuerzo computacional exigido para lograr los pasos requeridos en el algoritmo de Shor. Por ello es que se puede reemplazar y por $2L$ (ignorando el hecho que $2L$ no necesita ser entero). Estimar la diferencia entre $[\log_2(N^2)] \cong 2L$ y $[\log_2(2N^2)] \cong 2L+1$ es despreciable, ya que el esfuerzo computacional exigido para lograr los pasos individuales que constituyen el algoritmo de Shor no depende de lo que se prefiera, si lo propuesto por Shor o por Williams y Clearwater para estimar el valor $j_{\text{máx}}$ requerido, podemos concluir que para N grande el valor requerido de $j_{\text{máx}}$ es más pequeño que el propuesto por Shor por lo que se elije la estimación de Shor. Para determinar r y con ello factorizar un número grande, se requerirá una computadora cuántica con no menos de aproximadamente $3L$ qbits. En otros términos, factorizar un número recomendado RSA-309 que corresponde a 1024 dígitos binarios aparentemente requeriría una computadora cuántica de por lo menos 3072 qbits.

La primer carga registrada con los enteros menores o iguales a $2^y - 1$
Después de ordenar y poner un subíndice en los qbits y del registro Y como se discute en las Ecuaciones (B.6)-(B.10), la función de onda puede escribirse como $|j\rangle_y$, donde el subíndice indica que es la función de onda para el registro Y y j un entero entre $0 \leq j \leq 2^y - 1$, que se escribirá con notación decimal. Cuando el registro Y está en el estado descrito por la función de onda base $|j\rangle_y$, la representación de j en notación binaria revela inmediatamente el estado base de un qbit, $|0\rangle_k$ o $|1\rangle_k$, de cada uno de los qbits en el registro Y. Se entiende que el qbit k $(1 \leq k \leq y)$, cuyos estados bases son identificados por el subíndice k, corresponde al k-ésimo dígito, numerando de derecha a izquierda. El cálculo de la función de onda base para el registro Z se denota similarmente por $|i\rangle_Z$ donde $0 \leq i \leq 2^z - 1$. Se postula que inicialmente cada uno de los qbits de $y + z$ que constituyen la computadora cuántica pueden colocarse en su propio estado base de 1-qbit $|0\rangle$, es decir, la función de onda inicial de la computadora cuántica completa es $\Psi_C^{(0)} = |0\rangle_Y |0\rangle_Z$, donde el subíndice C denota la función de onda de la computadora completa (ver la Figura 2.3).
Procediendo con el algoritmo se requiere transformar el registro inicial Y de la función de onda $\Psi_C^{(0)} = |0\rangle_Y$ para pasar a una segunda fórmula

$$\Psi_Y^{2S} = 2^{-y/2} \sum_{j=0}^{2^y - 1} |j\rangle_Y, \tag{2.15}$$

es decir, requiere reemplazar el $|0\rangle_Y$ inicial por la suma en el lado derecho de la Ecuación (2.15), en donde una medida del estado en el registro Y producirá cualquiera de los enteros entre 0 y $2^y - 1$. Hay $2^y |j\rangle_Y$ independientes en el lado derecho de la Ecuación (2.15). Así, el factor $2^{-y/2}$ garantiza que Ψ_Y^{2S} es normalizada. Como se sabe que $N^2 \leq 2^y - 1$, la suma en la Ecuación (2.15) incluye cada j menor o igual al $j_{\text{máx}} = N^2$ recomendado por

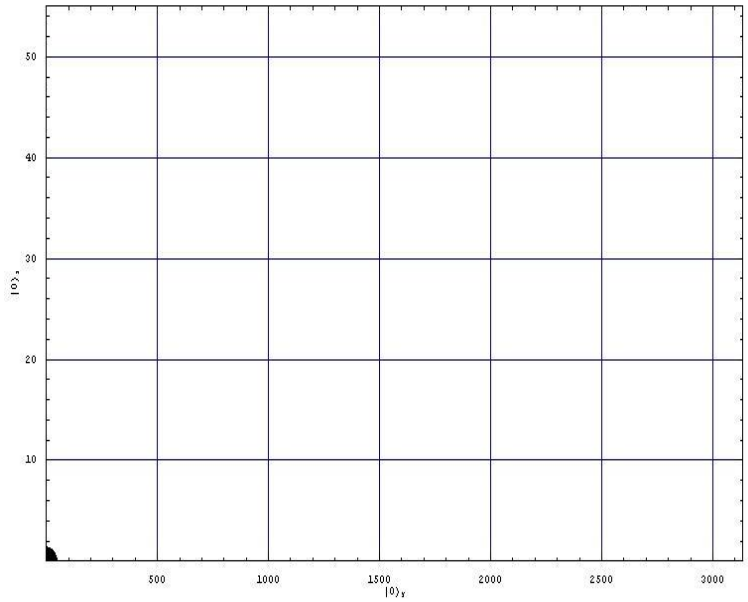

Figura 2.3: $\Psi_C^{(0)} = |0\rangle_Y |0\rangle_Z$

Shor.
La transformación de $|0\rangle_Y$ a Ψ_Y^{2S} de la Ecuación (2.15) se cumple por el uso de 1-qbit en la operación $\hat{U}_H$ conocida como **transformación de Hadamard**, y la operación de Hadamard en 1-qbit en el estado de la función de onda base $|0\rangle$ y $|1\rangle$ se define como:

$$\hat{U}_H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad \hat{U}_H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{2.16}$$

$\hat{U}_H$ se conoce por ser unitario. El factor de $\frac{1}{\sqrt{2}}$ en la Ecuación (2.16) permite que $\hat{U}_H$ conserve la normalización, cuando tiene una operación unitaria, la operación de Hadamard que se realiza sólo en el qbit k, sin afectar otros qbits, se denota por $\hat{U}_{Hk}$. Luego, consideremos el resultado de operar con $\hat{U}_{Hk}$ en una computadora con función de onda base $|j\rangle_Y$ para que el k-ésimo dígito (numerando de derecha a izquierda, será 0) en la representación binaria de j, significa que el producto de 1-qbit en la función de onda base que constituyen $|j\rangle_Y$ incluye el factor de $|0\rangle_k$ (no $|1\rangle_k$). Para obtener el resultado deseado, necesitamos sólo poner el subíndice k en cada una de las funciones de onda base en la Ecuación (2.16). Es más, como nuestro $|j\rangle_Y$ presente no contiene ningún estado base $|1\rangle_k$, sólo nos interesa la primera igualdad en la Ecuación (2.16). Se sigue que, excepto por el factor $\frac{1}{\sqrt{2}}$, la operación de $\hat{U}_{Hk}$, en $|j\rangle_Y$ se reemplaza $|0\rangle_k$ en $|j\rangle_Y$ por $|0\rangle_k + |1\rangle_k$, mientras que se deja igual $|j\rangle_Y$. En la expansión binaria del entero j, sin embargo, cambiando el k-ésimo dígito de 0 por 1 (siempre leyendo de derecha a izquierda) produce la expansión

binaria del entero $j + 2^{k-1}$. Por consiguiente, cuando el $|j\rangle_Y$ no contiene el estado base $|1\rangle_k$,

$$\hat{U}_{Hk}|j\rangle_Y = \frac{1}{\sqrt{2}}[|j\rangle_Y + |j + 2^{k-1}\rangle_Y]. \tag{2.17}$$

Ahora realizamos las y operaciones de $\hat{U}_{H1}, \hat{U}_{H2}, \hat{U}_{H3}, \ldots, \hat{U}_{Hy}$ secuencialmente (primero $\hat{U}_{H1}$) en el registro inicial Y de la función de onda $|0\rangle_Y \equiv \Psi_Y^{(0)}$. Sabemos que el $|0\rangle_Y$ no contiene ningún $|1\rangle_i$ factor para cualquier i, $1 \leq i \leq y$. Así podemos emplear la Ecuación (2.17) para la primera de estas operaciones y obtener

$$\begin{aligned} \Psi_Y^{(1)} &= \hat{U}_{H1}|0\rangle = \frac{1}{\sqrt{2}}[|0\rangle_Y + |0 + 2^0\rangle_Y] \\ &= \frac{1}{\sqrt{2}}[|0\rangle_Y + |1\rangle_Y] \\ &= \frac{1}{\sqrt{2}}\sum_{j=0}^{1}|j\rangle_Y. \end{aligned} \tag{2.18}$$

Debido a que $\hat{U}_{H1}$ tiene que ser definido para que realice la operación de Hadamard sólo en el qbit 1, la función de onda $\Psi_Y^{(1)}$ (como $\Psi_Y^{(0)}$) no contiene el factor $|1\rangle_2$, como se evidencia directamente del hecho que ambos enteros 0 y 1 en el lado derecho de la Ecuación (2.18) sean menores que 2. Por consiguiente, podemos emplear la Ecuación (2.17) durante la segunda de estas operaciones secuenciales, de este modo el resultado para $\hat{U}_{H2}\hat{U}_{H1}|0\rangle = \hat{U}_{H2}\Psi_Y^{(1)} = \Psi_Y^{(2)}$ es

$$\begin{aligned} \Psi_Y^{(2)} &= \frac{1}{\sqrt{2}}\hat{U}_{H2}[|0\rangle_Y + |1\rangle_Y] = \frac{1}{2}[[|0\rangle_Y + |0 + 2\rangle_Y] \\ &\quad +[|1\rangle_Y + |1 + 2\rangle_Y]] \\ &= \frac{1}{2}\sum_{j=0}^{3}|j\rangle_Y. \end{aligned} \tag{2.19}$$

Como cada uno de los enteros en el lado derecho de la Ecuación (2.19) es menor que $4 = 2^2$, $\Psi_Y^{(2)}$ no contiene el factor $|1\rangle_3$, permitiendo usar la Ecuación (2.17) para evaluar $\hat{U}_{H3}\Psi_Y^{(2)}$. Procediendo de este modo, se ve que el resultado de la sucesión completa de operaciones de Hadamard en $|0\rangle_Y$ es

$$\Psi_Y^{(y)} = \hat{U}_{Hy}\hat{U}_{Hy-1}\cdots\hat{U}_{H2}\hat{U}_{H1}|0\rangle = 2^{-y/2}\sum_{j=0}^{2^y-1}|j\rangle_Y. \tag{2.20}$$

El lado derecho de la Ecuación (2.20) es lo que deseamos $\Psi_Y^{(2S)}$, es decir lo obtenido en la Ecuación (2.15). La Figura 2.4 representa el estado de la función al aplicarle la operación de Hadamard. La transformación del paso inicial $|0\rangle_Y$ a $\Psi_Y^{(2S)}$ no requiere más de $y = 2L$ puertas cuánticas universales en la operación de Hadamard.

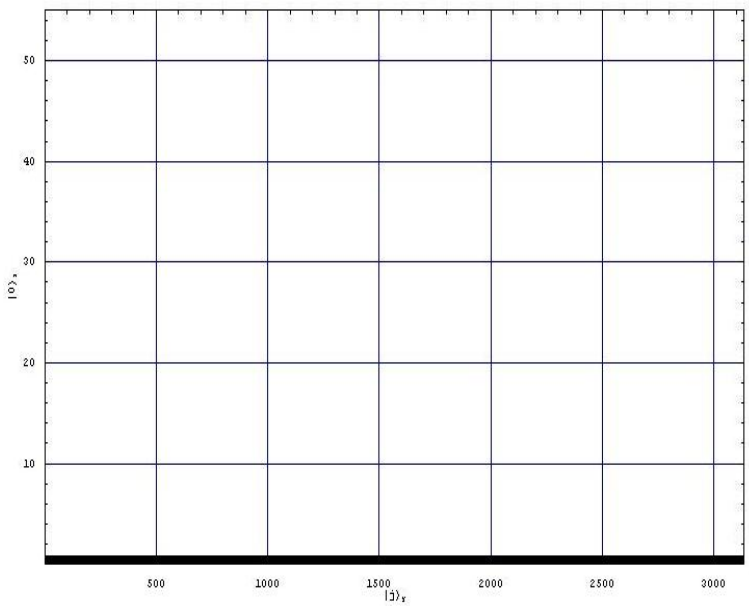

Figura 2.4: $\Psi_Y^{2S} = 2^{-y/2} \sum_{j=0}^{2^y-1} |j\rangle_Y$

§12.2. El cálculo de f_j

Después de completar el segundo paso, la función de onda es $\Psi_C^{2S} = \Psi_Y^{2S}|0\rangle_Z$, lo que significa que un estado de medida en el registro Z todavía garantiza que produce sólo el 0, independiente del valor de j que sea dado en un estado de medida simultáneo en el registro Y. Para el próximo paso se requiere n coprimo a N, con $1 < n < N$ o $1 < n < 2^y - 1$, y calculando el mcd entre n y N para comprobar que n es coprimo con N, pero la probabilidad de escoger un n al azar que no sea coprimo a N es tan pequeña que el esfuerzo de calcular el mcd no parece valer la pena. Si el entero n seleccionado no es coprimo a N, este hecho se pondrá claro cuando se verifique que el valor del orden supuesto r, no satisface la Congruencia (2.10). Por lo que será necesario repetir pasos y escoger otro n que sí sea coprimo a N.
El próximo paso del algoritmo es transformar Ψ_C^{2S} a su tercera fórmula

$$\Psi_C^{3S} = 2^{-y/2} \sum_{j=0}^{2^y-1} |j\rangle_Y |f_j\rangle_Z, \tag{2.21}$$

donde f_j se define por la Congruencia (2.8). En la función de onda Ψ_C^{3S} de la Ecuación (2.21), al realizar una medida sobre la colección de qbits en el registro Z producirá uno de los enteros $1 \leq f_j \leq N-1$. Es más, debido a la periodicidad de f_j, cada uno de los f_j, en la Ecuación (2.21) debe ser igual a uno de los (todos necesariamente diferentes) $f_1, f_2, \ldots, f_r = f_0 = 1$. Ningún otro entero puede ser el resultado de un estado de medida en el registro Z. En particular, como n es coprimo a N por definición, tal medida ahora posiblemente no puede producir el resultado 0.
En la Ecuación (2.21) cuando $N = 55$ y tomando $n = 12$ se obtiene $r = 4$ en la sucesión f_j (empezando con $j = 0$): $1, 12, 34, 23, 1, 12, 34, 23, 1, 12, 34, \ldots$. De acuerdo con este caso la Ecuación (2.21) es

$$\begin{aligned}\Psi_C^{3S} &= 2^{-y/2}[|0\rangle_Y|1\rangle_Z + |1\rangle_Y|12\rangle_Z + |2\rangle_Y|34\rangle_Z \\ &+ |3\rangle_Y|23\rangle_Z + |4\rangle_Y|1\rangle_Z + |5\rangle_Y|12\rangle_Z + |6\rangle_Y|34\rangle_Z \\ &+ |7\rangle_Y|23\rangle_Z + |8\rangle_Y|1\rangle_Z + |9\rangle_Y|12\rangle_Z + |10\rangle_Y|34\rangle_Z \\ &+ |11\rangle_Y|23\rangle_Z + |12\rangle_Y|1\rangle_Z + \cdots + |2^y - 1\rangle_Y \\ &\quad |f_{2^y-1}\rangle_Z]. \end{aligned} \tag{2.22}$$

La Ecuación (2.22) muestra la secuencia del registro Z en la función de onda base $|1\rangle_Z, |f_1\rangle_Z, |f_2\rangle_Z, \ldots, |f_{2^y-1}\rangle_Z$, la Ecuación (2.21) manifiesta la misma periodicidad r en la sucesión f_j, $0 \leq j \leq 2^y - 1$ (ver la Figura 2.5).

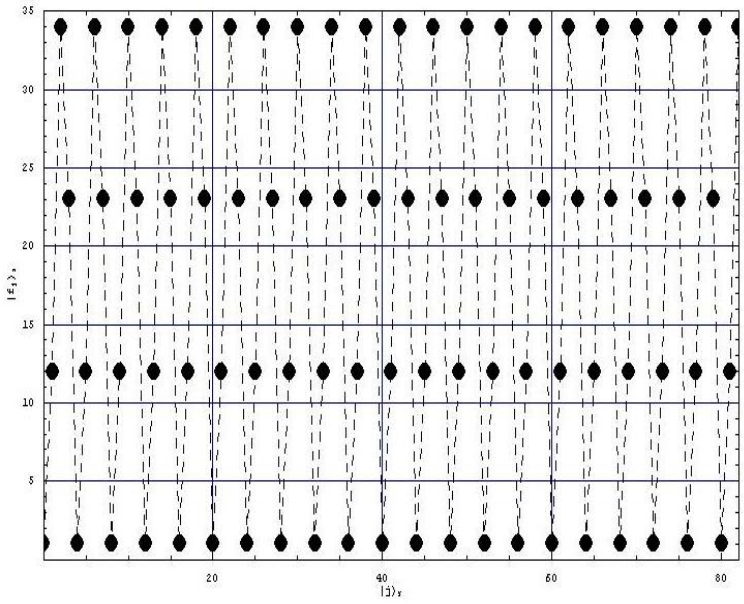

Figura 2.5: $\Psi_C^{3S} = 2^{-y/2} \sum_{j=0}^{2^y-1} |j\rangle_Y |f_j\rangle_Z$

§12.3. La medida en el estado de registro Z

Supongamos que la computadora se encuentra en el estado representado por Ψ_C^{3S} de la Ecuación (2.21). El objetivo de los próximos pasos en el algoritmo es extraer el valor r de la periodicidad que se presenta en Ψ_C^{3S}. Antes de realizar la medida en el registro Z, no tenemos idea de qué valores de f_j se encuentran en la Ecuación (2.21). Es más, al realizar cualquier medida en el registro Z, revelará uno de los valores de f_j que aparecen en la Ecuación (2.21), y automáticamente destruirá toda la información sobre los otros valores de f_j. No obstante, el paso siguiente en el algoritmo es medir el estado del registro Z, por lo que suponemos que produce el valor particular de f_k (uno de todos los posibles valores $f_0 = 1, f_1, f_2, \ldots, f_{r-1}$). Entonces después de realizar la medida en el registro Z la función de onda, toma su cuarta fórmula

$$\Psi_Y^{4S} = Q^{-1/2} \sum_{b=0}^{Q-1} |k + br\rangle_Y, \tag{2.23}$$

la Ecuación (2.23) ha retenido sólo aquellos $|j\rangle_Y$ de la Ecuación (2.21) que multiplican a $|f_k\rangle_z$. Q es igual al número de términos en la Ecuación (2.21) que contienen el factor $|f_k\rangle_z$; el factor $Q^{-1/2}$ es necesario para garantizar que la función de onda Ψ_Y^{4S} de la Ecuación (2.23) sea normalizada.

Para comprender la estructura de la Ecuación (2.23) y para ver cómo se estima la magnitud de Q, regresemos a nuestro ejemplo $N = 55, n = 12, r = 4$. Supongamos que el resultado al realizar la medida en registro Z de la computadora en el estado representado por la Ecuación (2.22) es $f_j = 1$ (ver la Figura 2.6). Entonces, después de realizar la medida en el registro Z, la función de onda toma su cuarta fase de operación en el algoritmo.

$$\Psi_Y^{4S} = Q^{-1/2}[|0\rangle_Y + |4\rangle_Y + |8\rangle_Y + \ldots + |4(Q-1)\rangle_Y] \tag{2.24}$$

Evidentemente, la medida en el registro Z de la Ecuación (2.21) ha propiciado la dependencia de r en la Ecuación (2.23), es decir, ha transformado el registro Y en una progresión aritmética (con la diferencia común r) de los enteros $j = k + br$ en el índice computacional de la función de onda base $|j\rangle_Y$. El valor de Q en la Ecuación (2.23) es determinado ya que $k + r\,(Q-1)$ no excede $2^y - 1$. Como $0 \leq k < r$ y Q es un entero por la definición, esta condición implica

$$Q = [r^{-1}(2^y - 1 - k)] + 1. \tag{2.25}$$

§12.4. Transformada de Fourier cuántica en la operación del registro Y de la función de onda

El valor de r se extrae finalmente de Ψ_Y^{4S} de la Ecuación (2.23) vía la operación de transformada de Fourier cuántica $\hat{U}_{FT}$. La operación $\hat{U}_{FT}$ transforma cualquier estado $|j\rangle_Y$ en el registro Y en

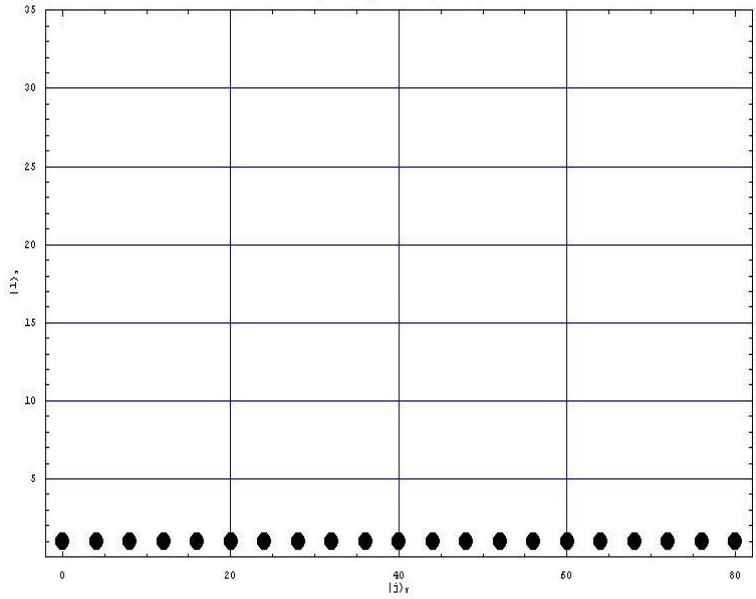

Figura 2.6: $\Psi_Y^{4S} = Q^{-1/2} \sum_{b=0}^{Q-1} |k + br\rangle_Y$

$$\hat{U}_{FT}|j\rangle_Y = 2^{-y/2} \sum_{c=0}^{2^y-1} e^{2\pi ijc/2^y} |c\rangle_Y. \tag{2.26}$$

Después de la operación $\hat{U}_{FT}$, la función de onda para el registro Y toma su quinta forma

$$\Psi_Y^{5S} = \hat{U}_{FT}\Psi_Y^{4S} = (2^y Q)^{-1/2} \sum_{c=0}^{2^y-1} \sum_{b=0}^{Q-1} e^{2\pi i(k+br)c/2^y} |c\rangle_Y. \tag{2.27}$$

En la Ecuación (2.27) el coeficiente dado de cualquier $|c\rangle_Y$ es una serie geométrica y se puede sumar para dar

$$\begin{aligned} \Psi_Y^{5S} &= (2^y Q)^{-1/2} \sum_{c=0}^{2^y-1} e^{2\pi ikc/2^y} \frac{1 - e^{2\pi ircQ/2^y}}{1 - e^{2\pi irc/2^y}} |c\rangle_Y \\ &= (2^y Q)^{-1/2} \sum_{c=0}^{2^y-1} e^{2\pi ikc/2^y} e^{\pi irc(Q-1)/2^y} \\ &\times \frac{\text{sen}(\pi rcQ/2^y)}{\text{sen}(\pi rc/2^y)} |c\rangle_Y. \end{aligned} \tag{2.28}$$

§12.5. El estado de medida del registro de Y

Si se realiza una medida en el registro Y se encontrará un estado particular de $|c\rangle_Y$. La probabilidad P_c de encontrar el estado $|c\rangle_Y$ es la norma al cuadrado del coeficiente de $|c\rangle_Y$ en la Ecuación (2.28), es decir,

$$P_c = (2^y Q)^{-1} \frac{\text{sen}^2(\pi r c Q/2^y)}{\text{sen}^2(\pi r c/2^y)}. \tag{2.29}$$

La operación $\hat{U}_{FT}$ no involucra el registro Z, al igual que la Ecuación (2.29) para la probabilidad de encontrar en el registro Y el estado $|c\rangle_Y$. Por ello es preferible medir los estados de los registros en dos pasos separados.
De la Ecuación (2.29), si consideramos que el orden r es una potencia de 2, entonces Q es exactamente igual a $2^y/r$. Por lo que, la Ecuación (2.29) se vuelve

$$P_c = (2^y Q)^{-1} \frac{\text{sen}^2 \pi c}{\text{sen}^2(\pi r c/2^y)}. \tag{2.30}$$

Como c es un número entero $0 \leq c \leq 2^y - 1$, la Ecuación (2.30) implica $P_c = 0$ para cualquier c; pero si se llegara a tomar algún valor de c que cumpla $d = rc/2^y$ con d un número entero, como puede ocurrir porque $2^y/r$ ya es un número entero, esos valores excepcionales de c son los que satisfacen

$$\frac{c}{2^y} - \frac{d}{r} = 0, \tag{2.31}$$

por lo que el lado derecho de la Ecuación (2.30) se vuelve $0/0$, y tenemos que regresar a la Ecuación (2.27), donde vemos eso excepto el factor común $e^{2\pi i k/2^y}$, cada término de b en la suma es dado por c. El número de términos en la suma es Q. Cuando r es una potencia par y el registro de Y está en el estado descrito por la función de onda Ψ_Y^{5S} de la Ecuación (2.27), la probabilidad P_c de que el registro Y se encuentre en el estado base $|c\rangle_Y$ es cero excepto cuando c satisface la Ecuación (2.31) la probabilidad es $P_c = (2^y Q)^{-1} Q^2 = 1/r$. Es más, debido a que $c < 2^y$, los únicos valores de c que pueden observarse son los enteros d en el rango $0 \leq d < r$. Así la probabilidad total de observar estos valores de c es $rP_c = 1$, como debe ser.
Considérese la circunstancia más general en que el orden r no es una potencia de 2. En este caso no existen valores de c que satisfagan la Ecuación (2.31) para cada entero d tal que $0 \leq d < r$. De hecho, si r es impar, no hay valores de c que satisfagan la Ecuación (2.31). Además de que $Q - 2^y/r$ ahora es igual a ξ, donde $-1 < \xi < 1$. Cuando r no es una potencia de 2, el numerador en la Ecuación (2.29) no es cero excepto posiblemente por un limitado número de valores muy especiales de c. En otros términos, para la mayoría, quizás todos, los valores de c, P_c ahora no es cero. No obstante, para cada entero d en el rango permitido, la probabilidad de observar el resultado c en una medida del registro de Y permanece grande para estos valores excepcionales de c, aunque ya no satisfaciendo la Ecuación (2.31). Para

cuantificar esta afirmación, note primero que como $r < N/2$, el máximo valor permitido de d/r (a saber $1-1/r$) ciertamente es menor que el máximo valor permitido de $c/2^y$ ($1-1/2^y$ es mayor que $1-1/N^2$). Debido a que el espacio entre la sucesión de valores de $c/2^y$ es 2^{-y}, cada valor permitido de d/r satisface exactamente la Ecuación (2.31) para algún valor de c u otro diferente a $c/2^y$ pero que no sea mayor a $2^{-y}/2$. En otros términos, cuando r no es una potencia de 2, la Ecuación (2.31) se reemplaza por

$$|\frac{c}{2^y} - \frac{d}{r}| \leq 2^{-y}/2, \tag{2.32}$$

con la convicción de que para cada valor permitido de d/r, existe un solo $c = c_1$ que satisface la Ecuación (2.32) (excepto cuando la igualdad se da para un c_1, en el caso de que se dé la igualdad para un segundo $c = c_2 = c_1 \pm 1$, correspondiente a d/r queda precisamente a la mitad entre dos valores sucesivos de $c/2^y$). Por consiguiente, cuando c satisface la Ecuación (2.32), tenemos

$$\frac{c}{2^y} = \frac{d}{r} + \varepsilon 2^{-y}, \tag{2.33}$$

donde $-\frac{1}{2} \leq \varepsilon \leq \frac{1}{2}$.
Si empleamos la Ecuación (2.33) en la Ecuación (2.29), la probabilidad de encontrar el registro Y en el estado $|c\rangle_y$ (cuando r no es una potencia de 2) es

$$P_c = (2^y Q)^{-1} \frac{\operatorname{sen}^2(\pi r \varepsilon Q/2^y)}{\operatorname{sen}^2(\pi r \varepsilon/2^y)} \geq \frac{Q}{2^y} \frac{\operatorname{sen}^2(\pi r \varepsilon Q/2^y)}{(\pi r \varepsilon Q/2^y)^2}, \tag{2.34}$$

hemos usado el hecho que $\operatorname{sen} x \leq x$ para $x \geq 0$. Como 2^y es muy grande comparado con $r < N/2$, en la estimación del lado derecho de la Ecuación (2.34) se da después el reemplazo de Q por $2^y/r$ (Q difiere de $2^y/r$ por una cantidad ξ, $|\xi| < 1$) para introducir un error. En otras palabras, el argumento de la función seno en el lado derecho de la Ecuación (2.34) puede tomarse por $\pi\varepsilon$. De la Ecuación (2.34) resulta.

$$P_c \geq r^{-1} \frac{\operatorname{sen}^2 \pi\varepsilon}{(\pi\varepsilon)^2} \geq r^{-1} \frac{1}{(\pi/2)^2} = \frac{4}{r\pi^2}, \tag{2.35}$$

donde la segunda desigualdad es el resultado de usar el hecho de que $\operatorname{sen} x/x$ es decreciente en el rango de $0 \leq x \leq \pi$, y reemplazando $|\xi|$ por su valor máximo permitido $1/2$. Como hay un c y un P_c asociado para cada uno los valores de r permitidos en d en la Ecuación (2.32), concluimos que, cuando r es una potencia de 2, la probabilidad total $P = rP_c$ de encontrar el registro Y en un estado $|c\rangle_y$ donde c satisface la Ecuación (2.32) no es menor a $4/\pi^2 \cong 0.4$.

§12.6. Determinando d/r

Después de haber obtenido c, que es, la medida del estado en el registro Y, todavía es necesario inferir el valor de r. Puede haber sólo una fracción d/r que satisface la

Ecuación (2.32) para c dado. Además d es un entero y $0 < d < r < N/2$. Para demostrar esto, note que si d_1/r_1 y d_2/r_2 son fracciones distintas, entonces

$$|\frac{d_1}{r_1} - \frac{d_2}{r_2}| = |\frac{d_1 r_2 - d_2 r_1}{r_1 r_2}| \geq \frac{1}{r_1 r_2} > \frac{4}{N^2}, \tag{2.36}$$

porque cuando $d_1/r_1 \neq d_2/r_2$, el entero $(d_1 r_2 - d_2 r_1)$ no puede ser igual a 0. Por otro lado, si d_1/r_1 y d_2/r_2 cada uno satisface la Ecuación (2.32) para el mismo c,

$$\begin{aligned} |\frac{d_1}{r_1} - \frac{d_2}{r_2}| &= |(\frac{c}{2^y} - \frac{d_2}{r_2}) - (\frac{c}{2^y} - \frac{d_1}{r_1})| \\ &\leq |\frac{c}{2^y} - \frac{d_2}{r_2}| + |\frac{c}{2^y} - \frac{d_1}{r_1}| \\ &\leq 2(2^{-y}/2) \\ &= 2^{-y} < \frac{1}{N^2}, \end{aligned} \tag{2.37}$$

Como las Ecuaciones (2.36) y (2.37) son inconsistentes, es imposible encontrar dos d/r distintos que satisfagan la Ecuación (2.32) para el mismo c.
Supongamos ahora que el estado de medida en el registro Y ha producido un estado $|c\rangle_y$ cuyo c satisface la Ecuación (2.32). La evaluación actual del único d/r en la Ecuación (2.32) se realiza extendiendo $c/2^y$ en una fracción continua, como Shor originalmente propuso. Un teorema importante es que si a/b es una fracción que satisface:

$$|\frac{a}{b} - x| < \frac{1}{2b^2}, \tag{2.38}$$

entonces a/b es un convergente de la fracción continua de x [6]. La Ecuación (2.32) tiene la forma de la Ecuación (2.38), con $x = c/2^y$ y $a/b = d/r$. Dado que $2^y > N^2$, el lado derecho de la Ecuación (2.32) es menor que $(2N^2)^{-1}$, que a su vez es menor a $(2r^2)^{-1}$, porque $r < N/2$. Este teorema implica que d/r debe ser un convergente de la fracción continua de $c/2^y$, es decir, expandiendo $c/2^y$ en una fracción continua producirá en algun momento a d/r como un convergente. Note que este resultado demuestra la importancia de escoger el tamaño y del registro Y. De hecho, si $2^y < N^2/4$, el lado derecho de la Ecuación (2.32) sería mayor que $2N^2$, y ya no aseguraría que d/r sea un convergente de la fracción continua de $c/2^y$. Similarmente, si $2^y < N^2/4$, el lado derecho de la Ecuación (2.37) sería mayor que $4/N^2$. Así, la Ecuación (2.37) ya no estaría en contradicción con la Ecuación (2.36), implicando que no se garantiza ahora ya que sólo haya un d/r permisible que satisfaga la Ecuación (2.32).
En seguida se muestra un ejemplo del método de fracciones continuas para determinar d/r. Al factorizar $N = 55$ por medio del algoritmo de Shor, en el registro Y se emplearán $y = 12$ qbits, $(2^{11} = 2048 < N^2 = 3025 < 2^y = 4096)$. El valor más grande posible de r es de 20 si se toma a $n = 37$. Para $d = 11$, el valor de d/r es exactamente 0.55. Tenemos

$\frac{2252}{4096} = 0.54980$, $\frac{2253}{4096} = 0.55005$, y $2^{-y}/2 = 0.00012$, es menor que $0.00020 = 0.55 - \frac{2252}{4096}$, pero mayor que $0.00005 = \frac{2253}{4096} - 0.55$. Si asumimos que la medida del estado en el registro Y ha producido $|c\rangle_Y$ consistente con la Ecuación (2.32) para $r = 20$ y $d = 11$, el valor de c debe ser 2253. Extendemos 2253/4096 en una fracción continua simple:

$$\frac{2253}{4096} = \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{4+\cfrac{1}{2+\cfrac{1}{50+\cfrac{1}{1+\frac{1}{3}}}}}}} \tag{2.39}$$

la fracción continua la podemos representar como $\frac{2253}{4096} = [0, 1, 1, 4, 2, 50, 3]$, ahora calculemos los convergentes de la fracción continua como se mostro en la Sección (§6):

$$\frac{m_0}{n_0} = \frac{0}{1} \qquad \frac{m_1}{n_1} = \frac{1}{1} \qquad \frac{m_2}{n_2} = \frac{1}{2} \qquad \frac{m_3}{n_3} = \frac{5}{9}$$

$$\frac{m_4}{n_4} = \frac{11}{20} \quad \frac{m_5}{n_5} = \frac{555}{1009} \quad \frac{m_6}{n_6} = \frac{566}{1029} \quad \frac{m_7}{n_7} = \frac{2253}{4096}$$

el convergente $\frac{m_4}{n_4} = \frac{11}{20} = \frac{d}{r}$ confirmando el teorema citado en el párrafo anterior. Es más, dado que $r < N/2$, en este ejemplo es $20 < \frac{55}{2}$, y no simplemente menor que N, de la Ecuación (2.38) aun cuando el lado derecho de la Ecuación (2.32) se había reemplazado por $2/N^2$, los valores de $c/2^y$ que satisfacen la Ecuación (2.32) habrían tenido un convergente igual a d/r. Pero $2^y > N^2$ implica $2/2^y < 2/N^2$. En otros términos, si una medida del estado en el registro Y produce un $|c\rangle_Y$ cuyo c satisface

$$|\frac{c}{2^y} - \frac{d}{r}| \leq 2(2^{-y}), \tag{2.40}$$

este $c/2^y$ también tendrá a d/r como un convergente de la fracción continua de $c/2^y$, aunque el valor de c no satisface la Ecuación (2.32). Por consiguiente, tenemos otra razón (además de la conveniencia de usar un promedio $|\varepsilon|$) para afirmar que la cantidad $4/\pi^2$ discutida en la Ecuación (2.35) en gran medida subestima la probabilidad de medir $|c\rangle_Y$ de los estados que puede producir d/r.

Para estimar esta probabilidad con más precisión, tómese en cuenta que si $c/2^y > d/r$ satisface la Ecuación (2.32), entonces cada uno de $c-2, c-1, c$, y $c+1$ satisface la Ecuación (2.40). Similarmente, si $c/2^y < d/r$ satisface la Ecuación (2.32), entonces $c-1, c, c+1$, y $c+2$ se requiere que cada uno satisfaga la Ecuación (2.40). En cualquier caso, agregando los cuatro P_c apropiados de la Ecuación (2.29), usando $\operatorname{sen} x \leq x$ para $x \geq 0$ como en la Ecuación (2.34), y aproximando Q por $2^y/r$ como se hizo derivando la

Ecuación (2.35), encontramos que la probabilidad P'_c de medir un estado $|c\rangle_Y$ que tendrá d/r como un convergente de la fracción continua $c/2^y$ es

$$P'_c \geq \frac{\operatorname{sen}^2 \pi\varepsilon}{\pi^2 r}\left(\frac{1}{(1+\varepsilon)^2} + \frac{1}{\varepsilon^2} + \frac{1}{(1-\varepsilon)^2} + \frac{1}{(2-\varepsilon)^2}\right), \tag{2.41}$$

donde $0 \leq \varepsilon \leq \frac{1}{2}$, y el primero en el P'_c indica que hemos sumado encima de los cuatro P_c apropiados. Para $\varepsilon = \frac{1}{2}$, obtenemos $P'_c = 80/9\pi^2 r = 0.90/r$; usando el promedio $\varepsilon = \frac{1}{4}$ obtenemos $P'_c = 0.935/r$.
Regresando al ejemplo de expansión de fracción continua de $\frac{2253}{4096}$ y verificar que tiene $\frac{11}{20}$ como un convergente, empleando la Ecuación (2.41) se puede estimar la probabilidad de inferir correctamente d/r al realizar una medida al estado $|c\rangle_Y$.

§12.7. Repetir los pasos para factorizar N

Inferir el valor de r no lleva inmediatamente a la factorización de N. La probabilidad de que r reuna los requisitos necesarios para poder factorizar N, es decir que r satisfaga la Congruencia (2.14), es aproximadamente $\frac{1}{2}$. Así aunque la probabilidad de poder inferir d/r vía una sola medida del registro Y es alrededor del 90 %, no obstante, será necesario volver a correr los pasos al menos dos veces en promedio antes de inferir d/r cuyo r pueda emplearse para factorizar N.

CAPÍTULO III

RESULTADOS

En este capítulo se darán a conocer los resultados obtenidos al realizar la implementación de algunos de los métodos discutidos en el capítulo anterior, para ello se han propuesto algunos números. Estos son producto de dos primos ya que el criptosistema RSA hace uso de tal tipo de números. Cada número ha sido factorizado registrando el tiempo que duró el cálculo con los distintos métodos.
La implementación de los métodos se realizó inicialmente en Wolfram Mathematica 8.0 pero el tiempo requerido para la factorización se mejoró al realizar la implementación en GP/PARI CALCULATOR Versión 2.5.0. Por lo que los resultados que se muestran fueron obtenidos al realizar el cálculo en PARI. Los algoritmos se encuentran en el anexo.
Debemos resaltar sin embargo, que las tablas mostradas sólo darán una burda idea de la eficiencia de los distintos métodos. Esto es así pues tal vez un número elegido sea más adecuado de factorizarse con un método que con otro. Para una mejor comparación sería necesario elegir una mayor cantidad de números.

Números propuestos

Los números N propuestos a factorizar fueron calculados con Wolfram Mathematica 8.0, cada número es producto de dos primos los cuales son desplegados al utilizar la instrucción **Prime**$[n]$, la cual calcula el n-ésimo número primo. La Tabla 3.1 muestra los números que se utilizarán para conocer la eficiencia (en tiempo) de algunos de los métodos discutidos en el capítulo anterior. La primera columna asigna nombre a cada número N, colocando RSA seguido de un guión y un número que corresponde a la cantidad de dígitos decimales que lo compone. La segunda y tercera columna muestra los factores primos por los que está compuesto el número N a factorizar, que se encuentra en la cuarta columna.

	p	q	N
RSA-10	641	6700417	$2^{2^5}+1$
RSA-12	16829	12467683	209818637207
RSA-14	64969	543391231	35303584886839
RSA-16	7823323	975298717	7630076884576591
RSA-18	536803	940312045661	504762327046961783
RSA-20	274177	67280421310721	$2^{2^6}+1$
RSA-22	1712573441	1612448540849	2761436546037200991409
RSA-24	15341653069	13415023448111	205808635653419085402659

Tabla 3.1: Números propuestos

Resultados

La implementación de los métodos en un principio se realizó con el software Wolfram Mathematica 8.0, realizando los cálculos de factorización en una computadora con las siguientes características:

- procesador: Intel(R) Core(TM)2 Duo CPU E4500 @ 2.20GHz 2.20GHz
- Memoria (RAM): 1.00 GB

Posteriormente se realizó la implementación en GP/PARI CALCULATOR Versión 2.5.0. en una computadora con las siguientes características:

- Dos Xeon con cuatro núcleos de 3.6GHz cada uno
- Memoria (RAM): 8.00 GB

Esta computadora se encuentra en el laboratorio de Criptografía de la Universidad Autónoma Metropolitana, Unidad Iztapalapa.
Con esta última implementación se lograron mejorar en gran medida los tiempos en que se consigue la factorización de los números propuestos, por ejemplo la Tabla 3.2 y la Figura 3.1 muestran los tiempos de factorización al utilizar el método de Fermat, en ellas se puede observar una gran diferencia entre los tiempos.

	p	q	N	Mathematica	PARI
RSA-10	641	6700417	$2^{2^5}+1$	13min, 34.309s	1,320ms
RSA-12	16829	12467683	209818637207	22min,39,720ms	2,473ms
RSA-14	64969	543391231	35303584886839	36h, 59min, 31,000ms	1min, 50,206ms
RSA-16	7823323	975298717	7630076884576591	57h, 37min, 42,000ms	2min, 48,382ms

Tabla 3.2: Tiempo de factorización con el método de Fermat

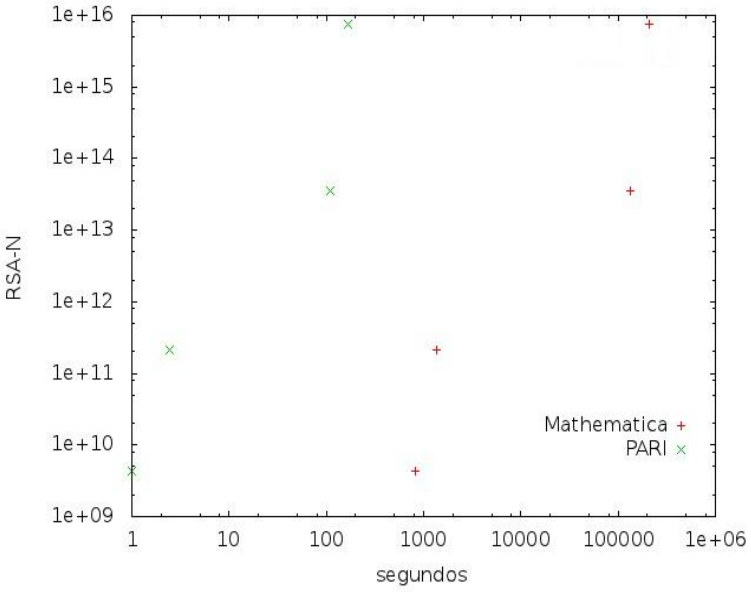

Figura 3.1: Factorización al utilizar el método de Fermat

En lo que resta de este capítulo se darán a conocer los resultados obtenidos con los diferentes métodos de factorización ejecutados en la segunda computadora antes mencionada con el software GP/PARI CALCULATOR Versión 2.5.0.

- **Método de Fermat**

  El método de Fermat no es muy complejo como se mostró en el capítulo anterior, pero sí es uno de los métodos que más tiempo requiere en la factorización de un número. Los resultados que se obtuvieron al factorizar los números propuestos N se muestran en la Tabla 3.3.

	Fermat
RSA-10	1,320ms
RSA-12	2,473ms
RSA-14	1min,50,206ms
RSA-16	2min, 48,382ms
RSA-18	55h, 48min, 2,178ms

Tabla 3.3: Factorización de N con el método de Fermat

Los números RSA-20, RSA-22 y RSA-24 no se lograron factorizar con este método,

con el número RSA-20 se paró el proceso después de 136h, 54min, 78,425ms de haber iniciado sin obtener la factorización de este número.

- **Método ρ de Pollard**

Al implementar el método ρ de Pollard se obtuvieron los resultados que se muestran en la Tabla 3.4, al factorizar los números propuestos N. Este método recordemos que hace uso de una función polinomial $f(x)$, para las factorizaciones mostradas se utilizó $f(x) = x^2 - 1$ y $f(x) = x^2 + 1$ con $x_0 = 2$.

ρ		
N	$x^2 - 1$	$x^2 + 1$
RSA-10	0ms	0ms
RSA-12	8ms	12ms
RSA-14	100ms	52ms
RSA-16	3,796ms	20,785ms
RSA-18	321ms	68ms
RSA-20	32ms	573ms
RSA-22	28min, 52,600ms	1h, 6min, 2,820ms
RSA-24	6h, 25min, 43,942ms	8h, 12min, 36,243ms

Tabla 3.4: Factorización de N con el método ρ de Pollard

- **Método $p - 1$ de Pollard**

En la implementación del método $p - 1$ de Pollard se hizo uso de una función de PARI llamada **prime(n)** la cual despliega el n-ésimo número primo, por ejemplo prime(2)=3. También se realizó una subrutina llamada exponenciación rápida la cual calcula a^m (mód N), y los resultados que se obtuvieron al factorizar los números propuestos se muestran en la Tabla 3.5. Cabe mencionar que para este método se propusieron números mayores que RSA-24 en su longitud, es decir, que estaban compuestos por más de 24 dígitos decimales, pero no se logró la factorización debido a que la función que se utiliza (prime(n)) tiene un límite, el cual es prime(41581)=500509.

$p-1$	
RSA-10	0ms
RSA-12	4ms
RSA-14	20ms
RSA-16	0ms
RSA-18	128ms
RSA-20	0ms
RSA-22	1,569ms
RSA-24	3,505ms

Tabla 3.5: Factorización de N con el método $p-1$ de Pollard

- **Método de fracciones continuas**

Con el método de fracciones continuas sólo se lograron factorizar los números que se muestran en la Tabla 3.6. Los números RSA-10 y RSA-20 no se logran factorizar debido a que se crea un ciclo al calcular los valores de b_i. El número RSA-24 no se logró factorizar a causa de que la memoria se desbordó por lo que el proceso fue detenido.

	FC	Cota	Precisión
RSA-12	40ms	500	200
RSA-14	468ms	750	600
RSA-16	300ms	750	500
RSA-18	3,532ms	1500	1750
RSA-22	18,906ms	3000	3000

Tabla 3.6: Factorización de N con el método de fracciones continuas

- **Método de curvas elípticas**

Al utilizar el método de curvas elípticas para factorizar los números propuestos se utilizó la curva $y^2 = x^3 + x + 1$ con un punto inicial (0,1) que pertenece a la curva, obteniendo los resultados que se muestran en la Tabla 3.7 logrando factorizar satisfactoriamente los números propuestos.

	CE
RSA-10	0ms
RSA-12	4ms
RSA-14	120ms
RSA-16	21,482ms
RSA-18	1,552ms
RSA-20	1,080ms
RSA-22	22min, 3,895ms
RSA-24	20h, 19min, 21,864ms

Tabla 3.7: Factorización de N con el método de curvas elípticas

- **Método de Dixon**

  El método de Dixon es muy parecido al de fracciones continuas, la diferencia es que el método de fracciones continuas hace uso del m-ésimo convergente y en el método de Dixon utilizamos $\lfloor \sqrt{i\,N} \rfloor$. Con este método se lograron factorizar los números propuestos N obteniendo los resultados que se muestran en la Tabla 3.8. Considerando una cota y precisión para cada número a factorizar cabe mencionar que si se coloca una precisión menor a la indicada, considerando la cota dada no se logra la factorización del número.

	Dixon	Cota	Precisión
RSA-10	132ms	100	120
RSA-12	252ms	500	220
RSA-14	2,849ms	750	425
RSA-16	8,881ms	750	1000
RSA-18	34,006ms	750	1100
RSA-20	6min, 34,657ms	1000	1150
RSA-22	7min, 44,286ms	1000	1350
RSA-24	27min, 258ms	1250	1750

Tabla 3.8: Factorización de N con el método de Dixon

CONCLUSIONES

La Tabla 3.9 muestra los registros (en tiempo) en que se logra la factorización de los números N con los distintos métodos, realizando los cálculos en GP/PARI CALCULATOR Version 2.5.0. Los espacios que aparecen en blanco en dicha tabla indican que no se logró la factorización del número con el método correspondiente, debido a que se crea un ciclo al ir calculando algunos valores dentro del algoritmo o que la memoria se desbordó por lo que el proceso fue detenido.

El método de Fermat es eficiente cuando los factores primos no se encuentran separados, es decir que los factores primos en su representación binaria tienen el mismo número de dígitos. En otro caso el método requiere de mucho tiempo para lograr la factorización comparándolo con los otros métodos, como se puede apreciar en la Tabla 3.9.

El método de fracciones continuas y el método de Dixon son eficientes pero son los que más memoria requieren, ya que crean una matriz enorme compuesta de ceros y unos.

El método más eficiente es el $p-1$ de Pollard, comparado con los otros métodos implementados, pero dicho método utiliza una función de GP/PARI CALCULATOR (**prime(n)**) la cual despliega el n-ésimo número primo. Pero esta función tiene un límite que es prime(41581)=500509, por lo que si se desea factorizar un número más grande al propuesto RSA-24 es posible que no se logre la factorización con la implementación realizada.

	p	q	N	Fermat	ρ	$p-1$
RSA-10	641	6700417	$2^{2^5}+1$	1,320ms	0ms	0ms
RSA-12	16829	12467683	209818637207	2,473ms	8ms	4ms
RSA-14	64969	543391231	35303584886839	1min,50,206ms	100ms	20ms
RSA-16	7823323	975298717	7630076884576591	2min, 48,382ms	3,796ms	0ms
RSA-18	536803	940312045661	504762327046961783	55h, 48min, 2,178ms	321ms	128ms
RSA-20	274177	67280421310721	$2^{2^6}+1$		32ms	0ms
RSA-22	1712573441	1612448540849	2761436546037200991409		28min, 52,600ms	1,569ms
RSA-24	15341653069	13415023448111	205808635653419085402659		6h, 25min, 43,942ms	3,505ms
	p	q	N	FC	CE	Dixon
RSA-10	641	6700417	$2^{2^5}+1$		0ms	132ms
RSA-12	16829	12467683	209818637207	40ms	4ms	252ms
RSA-14	64969	543391231	35303584886839	468ms	120ms	2,849ms
RSA-16	7823323	975298717	7630076884576591	300ms	21,482ms	8,881ms
RSA-18	536803	940312045661	504762327046961783	3,532ms	1,552ms	34,006ms
RSA-20	274177	67280421310721	$2^{2^6}+1$		1,080ms	6min, 34,657ms
RSA-22	1712573441	1612448540849	2761436546037200991409	18,906ms	22min, 3,895ms	7min, 44,286ms
RSA-24	15341653069	13415023448111	205808635653419085402659		20h, 19min, 21,864ms	27min, 258ms

Tabla 3.9: Tiempo de factorización con los distintos métodos

PERSPECTIVAS

La tecnología avanza hoy en día a pasos agigantados y la velocidad con la que se realizan los cálculos en un procesador se ha reducido en gran medida, sin embargo la factorización de números de gran longitud sigue siendo un problema dificil ya que no se cuenta con un método cuyo algoritmo pueda ser implementado en una computadora clásica para factorizar el número deseado en un tiempo factible, debido a esto el criptosistema RSA sigue siendo seguro a la práctica.

El algoritmo de Shor fue creado para ser ejecutado en una computadora cuántica pero el principal problema para la creación de una computadora cuántica es que la función de onda se colapsa antes de terminar los cálculos. Por lo que en caso de contar con una computadora cuántica con suficientes qbits el criptosistema RSA será obsoleto ya que la factorización del módulo sólo tomará unos minutos.

APÉNDICE A

PRUEBA DE PSEUDOPRIMO FUERTE

La prueba de pseudoprimo fue desarrollada por Pomerance, Selfridge, y Wagstaff en 1980. Los pseudoprimos son aquellos números que no siendo primos, verifican la prueba de base b, o lo que es lo mismo, sea $N \in \mathbb{Z}$, N es pseudoprimo respecto la base b si se verifica que son primos relativos, esto significa que N divide

$$b^{N-1} - 1.$$

Desde luego, podemos suponer que N es impar, de lo contrario no estaríamos perdiendo tiempo tratando de decidir si es primo. Se puede escribir

$$N = 2m + 1.$$

Así que N divide

$$b^{2m} - 1 = (b^m - 1)(b^m + 1),$$

Si N es primo, entonces debe dividir uno de los factores en el lado derecho, y no puede dividir a los dos, porque entonces sería dividir a su diferencia

$$(b^m + 1) - (b^m - 1) = 2.$$

Así que si N es en realidad un primo, entonces

$$b^m \equiv 1 \pmod{N} \quad \text{o} \quad b^m \equiv -1 \pmod{N}. \tag{A.1}$$

Por otro lado, si N es compuesto, hay una probabilidad razonable de que algunos de los factores que componen N divide b^m+1 mientras que otros dividen b^m-1. En este caso, N pasaría la prueba de pseudoprimo base b, pero no satisface la Congruencia A.1. Tomamos como ejemplo el primer pseudoprimo para la base 2:

$$341 = 11 \cdot 31.$$

En este caso $m = 170$. Y se calcula

$$2^{170} \equiv 1 \pmod{341}.$$

Pero no hemos terminado. Como el exponente es par, tenemos que

$$2^{170} - 1 = (2^{85} - 1)(2^{85} + 1).$$

Si 341 es primo, entonces tendríamos que

$$2^{85} \equiv 1 \pmod{341} \quad \text{o} \quad 2^{85} \equiv -1 \pmod{341}.$$

Pero no es asi ya que

$$2^{85} \equiv 32 \pmod{341}.$$

Lo que ha ocurrido es que 11 divide a $2^{85}+1$ y 31 divide a $2^{85}-1$.
En general, sea N primo relativo con b y paso la prueba de pseudoprimo módulo b. Escribe N como

$$N = 2^a \cdot t + 1,$$

donde t es impar y $a \geq 1$. entonces

$$b^{N-1} - 1 = (b^t - 1) \cdot (b^t + 1) \cdot (b^{2t} + 1) \cdot (b^{4t} + 1) \cdots (b^{2^{(a-1)}t} + 1), \tag{A.2}$$

y si N es primo, entonces divide exactamente uno de estos factores.

Definición: Un entero N impar se dice que es **pseudoprimo fuerte para la base** b si es compuesto, primo relativo con b y divide uno de los factores en el lado derecho de la Ecuación A.2.
Esto se puede poner en un algoritmo muy eficiente, que esencialmente se ejecuta muy rápido. Se conoce que N y b son primos relativos.

Algoritmo 15 Prueba de pseudoprimo fuerte

La ***entrada:*** $N \in \mathbb{Z}$ y b (b es cualquier entero positivo primo relativo con N).
La ***respuesta:*** paso la prueba o N falla.

PASO 1 Hacer $t = N - 1$, $a = 0$.
PASO 2 /$*$Encontrando t y a satisfaciendo: $N - 1 = 2^a \cdot t$, t impar.$*$/
Mientras t sea par hacer

- $t = t/2$
- $a = a + 1$

PASO 3 /$*$Checar $b^{2^i t} \pmod N$ para $i = 0$ hasta $a - 1$. Si alguna es correcta, entonces N pasa. De lo contrario, N falla.$*$/
$test = MODEXPO(b, t, N)$
Si $test = 1$ o $N - 1$ entonces **salida** paso la prueba y Fin.
Para $i = 1$ hasta $a - 1$ hacer

- $test = test \cdot test \pmod N$
- Si $test = N - 1$ entonces **salida** paso la prueba y Fin.

salida N fallo.

La prueba de pseudoprimo fuerte se implemento en GP/PARI CALCULATOR Versión 2.5.0. y el codigo se puede encontrar en el anexo.

Los primeros pseudoprimos para la base 2 son 341, 561, 645, 1105, 1387, 1729, 1905, 2047, 2465, 2701, ...
Los primeros pseudoprimos fuertes para la base 2 son 2047, 3277, 4033, 4681, 8321, 15841, 29341, 42799, 49141, 52633, 65281, 74665, 80581, 85489, 88357, 90751, ...

APÉNDICE B

LA FUNCIÓN DE ONDA

Los posibles resultados de la medida sobre el qbit son interpretados como corresponde a los enteros binarios 0 y 1 respectivamente, el par de funciones de onda ortogonales que describen los dos posibles estados de medida del qbit habitualmente se denotan por $|0\rangle$ y $|1\rangle$. Esta función de onda es conocida como la **base computacional**. La función de onda Ψ describe cualquier estado arbitrario del qbit, el cual es una superposición lineal de cualquier par de funciones de onda ortogonales; típicamente se expanden en términos de $|0\rangle$ y $|1\rangle$ como se muestra

$$\Psi = \mu|0\rangle + \nu|1\rangle \tag{B.1}$$

donde μ y υ son un par de números complejos que satisfacen

$$|\mu|^2 + |\nu|^2 = 1. \tag{B.2}$$

Una computadora cuántica es una colección de qbits, y por lo tanto éste es un sistema de mecánica cuántica cuyo estado debe ser descrito por una función de onda normalizada. Considerando, en particular, una computadora compuesta de sólo dos qbits, etiquetados por A y B. Entonces, habrá a lo más $2^2 = 2\times 2 = 4$ posibles resultados de las medidas sobre el par de qbits A, B (realizados simultáneamente o consecutivamente). Por consiguiente, la función de onda Ψ de esta computadora debe ser una superposición lineal de a lo más cuatro funciones de onda ortogonales de la base de 2-qbits.

$$\begin{aligned} |0\rangle_B|0\rangle_A &= |00\rangle, \\ |0\rangle_B|1\rangle_A &= |01\rangle, \\ |1\rangle_B|0\rangle_A &= |10\rangle, \\ |1\rangle_B|1\rangle_A &= |11\rangle. \end{aligned}$$

En otras palabras, la función de onda más general para la computadora de 2-qbit tiene la forma:

$$\Psi = \gamma_{00}|00\rangle + \gamma_{01}|01\rangle + \gamma_{10}|10\rangle + \gamma_{11}|11\rangle, \tag{B.3}$$

los dígitos binarios, de la función de onda base $|00\rangle, \ldots, |11\rangle$, se leen de derecha a izquierda y corresponden a los resultados de las medidas sobre los qbits B y A, respectivamente, donde las amplitudes asociadas $\gamma_{00}, \ldots, \gamma_{11},$, son números complejos que satisfacen

$$|\gamma_{00}|^2 + |\gamma_{01}|^2 + |\gamma_{10}|^2 + |\gamma_{11}|^2 = 1. \tag{B.4}$$

Los pares de dígitos 00, 01, 10, y 11, índices de las funciones de onda base que aparecen en la Ecuación (B.3), son la representación en sistema binario de los enteros 0 a 3 en el sistema decimal, con la condición de que cada representación binaria consiste de dos dígitos. Así, la Ecuación (B.3) puede reescribirse como

$$\Psi = \gamma_0|0\rangle + \gamma_1|1\rangle + \gamma_2|2\rangle + \gamma_3|3\rangle \tag{B.5}$$

donde las funciones de onda base $|i\rangle$ y sus amplitudes asociadas γ_i, $i = 0, 1, 2, 3$, son reetiquetadas, de las funciones de onda base $|00\rangle, \ldots, |11\rangle$ y de las amplitudes $\gamma_{00}, \ldots, \gamma_{11}$. Esto es ampliado para computadoras cuánticas más grandes, compuestas de $g > 2$ qbits. Puesto que un estado de medida sobre cualquier qbit puede tener a lo más dos resultados diferentes, el estado de medidas sobre la colección entera de qbits en una computadora cuántica de g-qbits puede tener a lo más 2^g resultados diferentes. La función de onda Ψ que describe algún estado de una computadora cuántica de g-qbits es una superposición lineal de a lo más 2^g funciones de onda base ortogonales de g-qbits. Si le ponemos índices a estos g-qbits, es decir, k corriendo de 1 a g, entonces las 2^g funciones de onda base pueden tomarse como

$$\begin{aligned} |0\rangle_g|0\rangle_{g-1}\ldots|0\rangle_2|0\rangle_1 &= |00\ldots00\rangle, \\ |0\rangle_g|0\rangle_{g-1}\ldots|0\rangle_2|1\rangle_1 &= |00\ldots01\rangle, \qquad \text{(B.6)} \\ \vdots \qquad\qquad & \qquad\quad \vdots \qquad \text{(B.7)} \\ |1\rangle_g|0\rangle_{g-1}\ldots|0\rangle_2|1\rangle_1 &= |10\ldots01\rangle \end{aligned}$$

y en analogía con la Ecuación (B.5) la función de onda más general para la computadora cuántica de g-qbits puede expresarse como

$$\Psi = \sum_{i=0}^{2^g-1} \gamma_i |i\rangle_g, \tag{B.8}$$

con

$$\sum_{i=0}^{2^g-1} |\gamma_i|^2 = 1. \tag{B.9}$$

En las Ecuaciones (B.8) y (B.9) los enteros i son escritos en el sistema decimal, como en la Ecuación (B.5); la representación binaria de cada i consiste de g dígitos. Cada función de onda de la base computacional $|i\rangle$ representa un estado del g-qbit, donde para todo k, $1 \leq k \leq g$, el resultado (0 ó 1) de un estado de medida sobre el k-ésimo qbit es igual al k-ésimo dígito (leyendo de derecha a izquierda) en la representación binaria de i; $|\gamma_i|^2$ es la probabilidad de encontrar la función de onda Ψ en el estado $|i\rangle_g$. Además, si la computadora se encuentra en el estado descrito por Ψ de la Ecuación (B.8), y un operador midiera, por ejemplo, los estados de qbits 1, 2 y g se obtendrían los resultados $|1\rangle_1$, $|0\rangle_2$ y $|1\rangle_g$, respectivamente. Estas medidas podrían colapsar Ψ en la función de onda $\Psi_M = \Psi_{g-3}[|1\rangle_1|0\rangle_2|1\rangle_g]$, donde la función de onda para $(g-3)$-qbits,

$$\Psi_{g-3} = [\sum_j |\gamma_j|^2]^{-1/2} \sum_j \gamma_j |j\rangle_{g-3}, \tag{B.10}$$

describe el estado de los restantes qbits 3, 4, ..., $(g-1)$ sabiendo que las medidas sobre los qbits 1, 2, y g en el sistema de g-qbits descrito por Ψ ha producido los resultados $|1\rangle_1$, $|0\rangle_2$, y $|1\rangle_g$ respectivamente. Además, j en la Ecuación (B.10) corre sobre todos los enteros cuya representación binaria de g-dígitos empieza con 1 y acaba con 01 (leyendo ahora de izquierda a derecha debido a la representación que se tiene de los qbits como se muestra en la Ecuación (B.6)), esto se debe a los resultados que se obtuvieron al realizar la medida en los qbits 1, 2, y g.

ANEXO

En este anexo aparecen los programas utilizados para realizar los diversos cálculos que aparecen a lo largo del escrito. Los programas estarán disponibles en el laboratorio de Criptografía del Departamento de Matemáticas, en la UAM-Iztapalapa.

Códigos de RSA

Generación de Claves

Este es un pequeño programa que calcula tanto las llaves públicas como las privadas en el criptosistema RSA.

```
p = Prime[27]; Print["p = ", p];
q = Prime[46]; Print["q = ", q];
n = p*q; Print["n = ", n];
FactorInteger[n];
\[Phi] = EulerPhi[n]; Print["\[Phi] = ", \[Phi]];
e = Random[Integer, {1, \[Phi]}];
e = 8207; Print["e = ", e];
mcd = GCD[e, \[Phi]]; Print["mcd(e,\[Phi]) = ", mcd];
d = PowerMod[e, -1, \[Phi]]; Print["d = ", d];
```

Código de cifrado

Este código fue utilizado para cifrar el mensaje del Ejemplo I.2, se puede utilizar para cifrar cualquier mensaje y si se quiere utilizar otra clave sólo hay que modificarlo un poco.

```
n = 20497; e = 8207;
abc = CharacterRange["a", "z"]; ABC = CharacterRange["A", "Z"];
T = Characters["hombresneciosqueacusaisalamujersinrazonsinverquesoislaocasi
ondelomismoqueculpaissorjuanainesdelacruz"];
(*T = Characters[InputString["Introduce el mensaje que quieres enviar"]];*)
(*se asigna un número a cada letra del texto claro*)
Te = {};
For[i = 1, i <= Length[T],
    For[j = 1, j <= Length[abc],
      If[T[[i]] == abc[[j]],AppendTo[Te,j - 1];, b];
    j++];
i++];  (*Print[Te];*)
(*Es separado en bloques de tres*)
Teto = {};
For[r = 0, r <= Length[Te] - 1,
  w = {};
    For[s = 1, s <= 3,
       If[r + s > Length[Te], AppendTo[w, 0];, AppendTo[w,Te[[r +s]]];];
       s++];
    AppendTo[Teto, w];
r = r + 3];  (*Print[Teto];*)
y = {};
For[j = 1, j <= Length[Teto],
      claro = Sum[ Teto[[j,i]]*26^(3 - i),{i,1,3}];
      AppendTo[y,claro];
j++];
(*El texto es cifrado y cambiado a base 26*)
claro = {};
claro1 = {};
For[i = 1, i <= Length[y],
    dky = PowerMod[y[[i]], e, n];
    AppendTo[claro1, dky];
    AppendTo[claro, IntegerDigits[dky, 26, 4]];
i++];  (*Print[claro1]; Print[claro];*)
Teto = {};
For[r = 1, r <= (Length[claro]),
   For[s = 1, s <= 4,
      AppendTo[Teto,claro[[r,s]]];
      s++];
r++];  (*Print[Teto];*)
(*Se sustituye por la letra correspondiente*)
```

```
  mecla = {};
  For[i = 1, i <= Length[Teto],
    AppendTo[mecla,ABC[[Teto[[i]] + 1]]];
    i++]; (*Print[mecla];*)
  mecla = StringJoin[mecla]; Print[mecla];
```

Código de descifrado

Para realizar el descifrado sólo hay que realizar unas pequeñas modificaciones en el código anterior y queda de la siguiente forma.

```
n = 20497; d = 3455;
abc = CharacterRange["a", "z"]; ABC = CharacterRange["A", "Z"];
T = Characters["AIIKAUFBAYQAACAXAHJEAEXJADMYAQJPAEOCAGHRAMBEARSHAENLAFVYAVERA
IJAAQJYAGKUAZKWAEDMATAGAPONAIYCAQKXBALQAWIDAIUPATONANVUALPEAWEYATIPAUPOACMO"];
(*T = Characters[InputString["Introduce el mensaje que quieres enviar"]];*)
(*se asigna un número a cada letra del texto claro*)
Te = {};
For[i = 1, i <= Length[T],
    For[j = 1, j <= Length[ABC],
      If[T[[i]] == ABC[[j]],AppendTo[Te,j - 1];, b];
    j++];
i++];  (*Print[Te];*)
(*Es separado en bloques de tres*)
Teto = {};
For[r = 0, r <= Length[Te] - 1,
  w = {};
    For[s = 1, s <= 4,
       If[r + s > Length[Te], AppendTo[w, 0];, AppendTo[w,Te[[r +s]]];];
       s++];
    AppendTo[Teto, w];
r = r + 4];  (*Print[Teto];*)
y = {};
For[j = 1, j <= Length[Teto],
      claro = Sum[ Teto[[j,i]]*26^(4 - i),{i,1,4}];
      AppendTo[y,claro];
j++];
(*El texto es cifrado y cambiado a base 26*)
claro = {};
claro1 = {};
For[i = 1, i <= Length[y],
```

```
    dky = PowerMod[y[[i]], d, n];
    AppendTo[claro1, dky];
    AppendTo[claro, IntegerDigits[dky, 26, 3]];
i++];  (*Print[claro1]; Print[claro];*)
Teto = {};
For[r = 1, r <= (Length[claro]),
   For[s = 1, s <= 3,
      AppendTo[Teto,claro[[r,s]]];
      s++];
r++];  (*Print[Teto];*)
(*Se sustituye por la letra correspondiente*)
  mecla = {};
  For[i = 1, i <= Length[Teto],
    AppendTo[mecla,abc[[Teto[[i]] + 1]]];
    i++]; (*Print[mecla];*)
  mecla = StringJoin[mecla]; Print[mecla];
```

Implementación de los métodos de factorización

La implementación de los métodos se realizó en PARI y a continuación se muestra el código fuente de los distintos algoritmos implementados.

Prueba pseudoprimo fuerte

```
pseudo(b,n)={
t=n-1;
a=0;
while((t%2)==0,
      t=t/2;
      a=a+1;
     );
test=fun(b,t,n);
if(test==1 || test==n-1,
   print("Paso la prueba");
   break(1);
  );
for(i=1,a-1,
    test=(test*test)%n;
    if(test==n-1,
       print("Paso la prueba");
       break(2);
```

```
      );
   );
print("N fallo");
}
```

Método de Fermat

```
fermat(n)={
x=truncate(sqrt(n))+1;
para=0;
while(para==0,
      if(issquare(x^2-n)==1,
         print("un factor es: ",truncate(x-sqrt(x^2-n)));
         print("un factor es: ",truncate(x+sqrt(x^2-n)));
         para=1;
        ,
         x++;
        );
     );
}
```

Método ρ de Pollard

```
rho(n)={
x=[2];
p=1;
i=2;
while(p==1,
      x=concat(x,[(x[i-1]^2-1)%n]);
      j=1;
      while(j<i,
            p=gcd(x[i]-x[j],n);
            if(p==1,
               j++;
              ,
               j=i;
              );
           );
      i++;
     );
print("un factor es: ",p);
```

```
print("un factor es: ",truncate(n/p));
}
```

Método $p-1$ de Pollard

```
pmenos1(n)={
B=truncate(sqrt(n));
a=3;
i=1;
q=2;
while(q<=B,
      q=prime(i);
      l=truncate((log(n))/(log(q)));
      m=q^l;
      a=fun(a,m,n);
      d=gcd(a-1,n);
      if(d==1 || d==n,,break);
      i++;
     );
print("un factor es: ",d);
print("un factor es: ",truncate(n/d));
}

\\exponenciacion rapida

fun(a,m,n)={
/*erm= a^m mod n   restricciones: 0<=a<n, n>=2, m>=0*/
erm=1;
x=a%n;
while(m>0,
      if((m%2)==1,
         erm=(erm*x)%n;
         x=(x*x)%n;
         m=(m-1)/2;
        ,
         x=(x*x)%n;
         m=m/2;
        );
     );
return(erm);
}
```

Método de fracciones continuas

```
\\farcciones continuas
fracc(n)={
x=sqrt(n);
nmi=900000;
c=500;
fb=[-1];
j=1;
while(fb[j]<c,
      fb=concat(fb,[prime(j)]);
      j++;
     );
i=1;
bis=[];
mis=[];
vecai=[;];
while(i<=nmi,
      cx=contfrac(x,i,i+1);
      m=contfracpnqn(cx)[1,1];
      b=(m^2)%n;
      if(isprime(b)==1,
         if(b>c,
            b=b-n;
           );
        ,
         facb=factor(b);
         con=0;
         for(va=1,matsize(facb)[1],
                  if(facb[va,1]>c,
                     con++;
                    );
            );
         if(con>0,
            b=b-n;
           );
        );

        if(isprime(b)==1,
         if(b>c,
            a=vector(length(fb),j,0);
```

```
            ,
             a=vector(length(fb),j,0);
             for(va=1,length(fb),
                 if(b==fb[va],
                    a[va]=1;
                   );
                );
            );
          ,
           facb=factor(b);
           con=0;
           for(va=1,matsize(facb)[1],
                    if(facb[va,1]>c,
                       con++;
                      );
              );
           if(con>0,
              a=vector(length(fb),j,0);
            ,
              a=vector(length(fb),j,0);
              for(z=1,matsize(facb)[1],
                  for(va=1,length(fb),
                      if(facb[z,1]==fb[va],
                         a[va]=facb[z,2]%2;
                        );
                     );
                );
            );
        );
       if(a!=0,
          vecai=concat(vecai,a~);
          bis=concat(bis,b);
          mis=concat(mis,m);
        ,
        );
     nulmod2=lift(matker(vecai*Mod(1,2)));
\\     print(b,"   ",m,a);
\\     print(fb,vecai);
\\     print(nulmod2);
     bibis=1;
     mimis=1;
```

```
     for(j=1,matsize(nulmod2)[2],
         for(i=1,matsize(nulmod2)[1],
             if(nulmod2[i,j]==1,
                bibis=bibis*bis[i];
                mimis=mimis*mis[i];
               );
            );
        );
     p=gcd((truncate(sqrt(bibis))%n)-(mimis%n),n);
     if(p!=1 && p!=n,
\\         print(i);
\\         print(bis,mis,nulmod2);
        print("un factor es: ",p);
        print("un factor es: ",truncate(n/p));
        break;
       ,
       );
      i++;
     );
}
```

Método de curvas elípticas

```
\\ curvas elipticas
ECM(n)={
a=1;
nmi=20000000000;
x1=0;
y1=1;
lan=((3*x1^2+a)%n)*(((2*y1)^-1)%n);
x2=(lan^2-2*x1)%n;
y2=(lan*(x1-x2)-y1)%n;
c=3;
while(c<=nmi,
      lan=((y2-y1)%n)*((x2-x1)^-1%n);
      x3=(lan^2-x1-x2)%n;
      y3=(lan*(x1-x3)-y1)%n;
      p=gcd(x3,n);
      if(p>1 && p!=n,
         print("un factor es: ",p);
         print("un factor es: ",truncate(n/p));
```

```
         break;
        ,
         x2=x3;
         y2=y3;
         c++;
        );
     );
}
```

Método de Dixon

```
dixon(n)={
x=sqrt(n);
nmi=900000;
c=750;
fb=[-1];
j=1;
while(fb[j]<c,
      fb=concat(fb,[prime(j)]);
      j++;
     );
i=1;
bis=[];
mis=[];
vecai=[;];
while(i<=nmi,
      m=truncate(sqrt(i*n));
      b=(m^2)%n;
      if(isprime(b)==1,
         if(b>c,
            b=b-n;
           );
        ,
         facb=factor(b);
         con=0;
         for(va=1,matsize(facb)[1],
                  if(facb[va,1]>c,
                     con++;
                    );
            );
         if(con>0,
```

```
        b=b-n;
       );
    );

    if(isprime(b)==1,
     if(b>c,
        a=vector(length(fb),j,0);
       ,
        a=vector(length(fb),j,0);
        for(va=1,length(fb),
            if(b==fb[va],
               a[va]=1;
              );
           );
       );
     ,
     facb=factor(b);
     con=0;
     for(va=1,matsize(facb)[1],
              if(facb[va,1]>c,
                 con++;
                );
        );
     if(con>0,
        a=vector(length(fb),j,0);
       ,
        a=vector(length(fb),j,0);
        for(z=1,matsize(facb)[1],
            for(va=1,length(fb),
                if(facb[z,1]==fb[va],
                   a[va]=facb[z,2]%2;
                  );
               );
           );
       );
    );
  if(a!=0,
     vecai=concat(vecai,a~);
     bis=concat(bis,b);
     mis=concat(mis,m);
    ,
```

```
        );
     nulmod2=lift(matker(vecai*Mod(1,2)));
\\     print(b,"   ",m,a);
\\     print(fb,vecai);
\\     print(nulmod2);
     bibis=1;
     mimis=1;
     for(j=1,matsize(nulmod2)[2],
         for(i=1,matsize(nulmod2)[1],
             if(nulmod2[i,j]==1,
                bibis=bibis*bis[i];
                mimis=mimis*mis[i];
               );
            );
        );
     p=gcd((truncate(sqrt(bibis))%n)-(mimis%n),n);
     if(p!=1 && p!=n,
\\        print(i);
\\        print(bis,mis,nulmod2);
        print("un factor es: ",p);
        print("un factor es: ",truncate(n/p));
        break;
       ,
       );
      i++;
     );
}
```

BIBLIOGRAFÍA

[1] P. van Oorschot, A. Menezes and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[2] D. M. Bressoud. *Factorization and Primality Testing.* Springer, 1989.

[3] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[4] Carl Friedrich Gauss. *Disquisitiones Arithmeticae.* Lipsiae, 1801.

[5] E. Gerjuoy. Shor's factoring algorithm and modern cryptography, an illustration of the capabilities inherent in quantum computers. *Department of Physics, University of Pittsburgh, Pittsburgh. Pennsylvania 15260. Am. J. Phys.*, 73(6):521–540, june 2005.

[6] G. H. Hardy, E. M. Wright revised by D. R. Heath-Brown, and J. H. Silverman. *An Introduction to the Theory of Numbers.* Oxford, sixth edition, 2008.

[7] Manindra Agrawal, Neeraj Kayal and Nitin Saxena. Primes is in p. Annals of Mathematics,160, pp 781-793, 2004.

[8] Thorsten Kleinjung. Factorization of a 768-bit rsa modulus. pages 333–350, 2010. Disponible en http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.567.

[9] D. Hankerson, A. Menezes and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer, 2004.

[10] Brian Antony Murphy. *Polynomial selection for the number field sieve integer factorisation algorithm.* PhD thesis, 1999.

[11] Carl Pomerance. The quadratic sieve factoring algorithm. *Department of Mathematics University of Georgia*, LNCS **209**:169–182, 1985.

[12] R. J. Schoof. Quadratic fields and factorisation. Computational Methods in Number Theory, R. Tijdeman and H. Lenstra, eds. Mathematisch Centrum, Amsterdam, Tract 155, Part II, Amsterdam, pp 235-286, 1983.

[13] M. Seysen. *A Probabilistic Factorization Algorithm with Quadratic Forms of Negative Discriminant*. Math. Comp. vol. 48, No 178. (1987).

[14] Douglas R. Stinson. *Cryptography: Theory and practice*. Chapman and Hall/Crc, 3rd edition, 2006.

[15] C. P. Williams and S. H. Clearwater. *Explorations in Quantum Computing*. Springer, 1998.

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00076
Matrícula: 209381984

FACTORIZACION DE ENTEROS

En México, D.F., se presentaron a las 13:00 horas del día 20 del mes de julio del año 2012 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DRA. MARTHA RZEDOWSKI CALDERON
DR. JOSE NOE GUTIERREZ HERRERA
DR. MARIO PINEDA RUELAS

UNIVERSIDAD AUTÓNOMA METROPOLITANA
UNIDAD IZTAPALAPA
20 JUL. 2012
RECIBIDO
DIVISIÓN C.B.I. DIRECCIÓN

Bajo la Presidencia de la primera y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS (MATEMÁTICAS APLICADAS E INDUSTRIALES)

DE: LEONEL SERGIO CARRASCO PEREZ

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

Aprobar

Acto continuo, la presidenta del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

LEONEL SERGIO CARRASCO PEREZ
ALUMNO

REVISÓ

LIC. JULIO CESAR DE LARA ISASSI
DIRECTOR DE SISTEMAS ESCOLARES

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JOSE ANTONIO DE LOS REYES HEREDIA

PRESIDENTA

DRA. MARTHA RZEDOWSKI CALDERON

VOCAL

DR. JOSE NOE GUTIERREZ HERRERA

SECRETARIO

DR. MARIO PINEDA RUELAS