

# Introducción a la criptografía y sus aplicaciones

José Angel de Bustos Pérez  
Versión 1,1



# Prefacio

El origen de este documento fue un trabajo que tuve que entregar para una asignatura sobre criptografía durante mis últimos años de carrera. Mi idea es la de ampliar y/o corregir este documento, pero debido, fundamentalmente, a la falta de tiempo esa labor la ire realizando lentamente.

Este documento es una versión mejorada, corregida y ampliada. He tratado de hacer un libro más práctico y añadir temas que omití en la primera versión como:

1. El nuevo estándar **AES**.
2. Firmas y certificados digitales.
3. Software criptográfico.
4. Computación cuántica.
5. Un índice de abreviaturas.
6. Información sobre los RFC's relacionados con la criptografía.

Al igual que en la primera versión seguiré haciendo hincapié en los métodos matemáticos. En el caso de que no te interese la criptografía desde el punto matemático mi consejo es que te las saltes las demostraciones y únicamente te fijes en los resultados que nos ofrecen los teoremas matemáticos, si te son de utilidad y/o interés.

Cualquier sugerencia, critica y/o corrección será bienvenida para la mejora y ampliación de este documento.

Mi dirección de correo electrónico es [jdebustos@augcyl.org](mailto:jdebustos@augcyl.org). Siente libre de distribuir, siempre y cuando sea de forma gratuita y altruista, este documento. Queda expresamente prohibida la modificación y/o alteración de este documento sin la expresa autorización del autor del mismo.

© José Angel de Bustos Pérez, Agosto 2.001.

# Índice general

<b>Prefacio</b>	<b>III</b>
<b>I Introducción a la Criptografía</b>	<b>11</b>
<b>1. Introducción</b>	<b>13</b>
1.1. ¿Qué es la criptografía? . . . . .	13
1.2. Utilidades de la Criptografía . . . . .	14
1.3. Criptografía Clásica . . . . .	14
1.4. Criptografía Moderna . . . . .	15
1.5. Ataques a la Seguridad Criptográfica . . . . .	17
1.5.1. Ataque a partir del cifrado . . . . .	18
1.5.2. Ataque a partir del texto en claro . . . . .	18
1.5.3. Ataque a partir del texto en claro elegido . . . . .	18
1.5.4. Ataque a partir de texto en claro condicionado . . . . .	18
1.5.5. Ataque a partir de cifrado elegido . . . . .	18
1.5.6. Ataque a partir de cifrado condicionado . . . . .	18
1.5.7. Ataque con clave conocida . . . . .	19
1.5.8. Reutilización del protocolo . . . . .	19
1.5.9. Suplantación de personalidad . . . . .	19
1.5.10. Compilación de un diccionario . . . . .	19
1.5.11. Búsqueda exhaustiva . . . . .	19
1.5.12. Ataque mediante intromisión . . . . .	19
<b>2. Criptoanálisis</b>	<b>21</b>
<b>3. Sistemas Criptográficos</b>	<b>23</b>
3.1. Algunos Sistemas Criptográficos . . . . .	24
3.1.1. Simple sustitución . . . . .	24
3.1.2. Transposición de orden d . . . . .	24
3.1.3. Vigenere . . . . .	25

3.1.4. El cifrador de César . . . . .	26
3.2. Sistemas de Cifrado . . . . .	26
3.2.1. Cifrado en flujo . . . . .	26
3.2.2. Cifrado en bloque . . . . .	26
3.3. Transformaciones Lineales . . . . .	27
3.4. Transformaciones Afines . . . . .	30
<b>4. Complejidad Computacional</b>	<b>31</b>
4.1. Como calcular el Coste Computacional . . . . .	32
4.1.1. Operaciones a nivel de “bits” . . . . .	32
4.2. Tiempo Polinomial . . . . .	34
4.2.1. Máquinas de Turing y el problema P . . . . .	35
4.2.2. Problemas de tipo P . . . . .	35
<b>II Criptografía Clásica</b>	<b>37</b>
<b>5. Algoritmos Simétricos</b>	<b>39</b>
5.1. Sistemas de Encriptación con Estructura de Grupo . . . . .	39
5.2. El Algoritmo DES . . . . .	40
5.3. Descripción del DES . . . . .	40
5.4. Encriptación con DES . . . . .	41
5.4.1. Procedimiento para obtener $L'$ . . . . .	41
5.4.2. Procedimiento para obtener $R'$ . . . . .	41
5.5. Desencriptación con DES . . . . .	43
5.6. Debilidad de las claves en el DES . . . . .	43
5.7. Variantes del DES . . . . .	45
5.7.1. DES Múltiple . . . . .	45
5.7.2. DES con Subclaves Independientes . . . . .	46
5.7.3. DES Generalizado . . . . .	46
<b>III Breve introducción a la Teoría de Números</b>	<b>47</b>
<b>6. Los Números Primos</b>	<b>49</b>
6.1. Infinitud de los Números Primos . . . . .	49
6.2. Distribución de los Números Primos . . . . .	50
6.3. Factorización de Números Enteros . . . . .	52
6.4. Diferentes clases de Números Primos . . . . .	55
6.4.1. Números primos de Fermat . . . . .	55
6.4.2. Números primos gemelos . . . . .	55

6.4.3.	Números primos de Mersenne . . . . .	56
6.4.4.	Números primos fuertes . . . . .	56
<b>7.</b>	<b>Factorización y primalidad</b>	<b>57</b>
7.1.	El Problema de la Factorización . . . . .	57
7.2.	Algoritmos . . . . .	58
7.2.1.	La criba de Eratóstenes . . . . .	59
7.2.2.	Algoritmo de fuerza bruta . . . . .	59
7.2.3.	Algoritmo II . . . . .	60
7.2.4.	Comparación de los algoritmos . . . . .	61
7.3.	Algoritmos Probabilísticos . . . . .	62
7.3.1.	Segundo teorema de Fermat . . . . .	62
7.3.2.	Algoritmos APRCL y Atkin-Morain . . . . .	63
7.3.3.	Algoritmo de Lehmann . . . . .	64
7.3.4.	Algoritmo de Rabin-Miller . . . . .	65
7.3.5.	En la práctica . . . . .	65
<b>8.</b>	<b>La Función <math>\Phi</math> de Euler y Congruencias Clásicas</b>	<b>67</b>
8.1.	La Función $\Phi$ de Euler . . . . .	67
8.1.1.	Propiedades de la función $\Phi$ de Euler . . . . .	68
8.2.	Congruencias Clásicas . . . . .	69
8.2.1.	Congruencia de Euler . . . . .	70
8.2.2.	Congruencia de Fermat . . . . .	71
<b>9.</b>	<b>El Algoritmo de Euclides</b>	<b>73</b>
9.1.	Cálculo del M.C.D. mediante el Algoritmo de Euclides . . . . .	74
9.1.1.	Algoritmo de cálculo . . . . .	75
9.2.	Cálculo de Inversos en $\mathbb{Z}/n$ . . . . .	75
9.2.1.	Caracterización de los invertibles de $\mathbb{Z}/n$ . . . . .	76
9.2.2.	Ecuaciones diofánticas en $\mathbb{Z}$ . . . . .	77
9.2.3.	Cálculo de inversos con el algoritmo de Euclides . . . . .	79
<b>IV</b>	<b>Criptografía Moderna</b>	<b>81</b>
<b>10.</b>	<b>Funciones de un Sentido</b>	<b>83</b>
10.1.	El Logaritmo Discreto . . . . .	84
10.1.1.	El problema del Logaritmo Discreto . . . . .	85
10.1.2.	Algoritmo de Exponenciación Rápida . . . . .	86

<b>11.El Algoritmo RSA</b>	<b>89</b>
11.1. Elección de Claves en el Algoritmo RSA	89
11.1.1. Elección de la Clave Privada, $K_{Pv}$	90
11.1.2. Elección de la Clave Pública, $K_{Pb}$	90
11.1.3. Datos Privados	90
11.1.4. Datos Públicos	91
11.2. Descripción del Algoritmo	91
11.2.1. Procedimiento de Encriptación	92
11.2.2. Procedimiento de Desencriptación	92
11.3. Ejemplo Práctico del Algoritmo RSA	93
11.3.1. Procedimiento de Encriptación	93
11.3.2. Procedimiento de Desencriptación	94
11.4. Justificación Matemática del Algoritmo RSA	95
11.5. Seguridad del Algoritmo RSA	95
11.5.1. Algoritmo RSA y el Algoritmo de Shor	96
11.5.2. Claves Débiles en RSA	96
11.5.3. Claves Demasiado Cortas	97
11.5.4. Ataques de Intermediario	97
11.5.5. Ataques de Texto Plano Escogido	98
11.5.6. Ataques de Módulo Común	98
11.5.7. Firmas digitales y RSA	99
<b>V Criptografía con Curvas Elípticas</b>	<b>101</b>
<b>12.Curvas Elípticas</b>	<b>103</b>
12.1. Definición de Curvas Elípticas	103
12.2. Estructura de grupo de las Curvas Elípticas	105
12.3. Curvas Elípticas sobre Cuerpos Finitos	109
12.3.1. Orden de las Curvas Elípticas definidas sobre Cuerpos de Característica 2	110
12.3.2. Cálculos en Coordenadas Proyectivas	111
12.4. Como obtener los múltiplos de puntos de una Curva Elíptica	112
12.4.1. Procedimiento de Izquierda a Derecha	113
12.4.2. Funciones de Schoof	114
12.5. Sistema Criptográfico de Massey-Omura	115
12.5.1. Debilidades de este Sistema Criptográfico	116
12.6. Equivalencia entre puntos de una Curva Elíptica y Mensajes	118
12.6.1. Como encontrar los puntos de la curva	118
12.6.2. Como obtener el mensaje	120
12.7. El Logaritmo Elíptico	120

<i>ÍNDICE GENERAL</i>	5
-----------------------	---

12.8. Test de Primalidad con Curvas Elípticas . . . . .	122
12.8.1. Descripción práctica del Algoritmo . . . . .	123

<b>VI Apéndices</b>	<b>125</b>
---------------------	------------

<b>A. Alfabetos</b>	<b>127</b>
---------------------	------------

A.1. Alfabeto con 29 símbolos . . . . .	128
A.2. Alfabeto con 36 símbolos . . . . .	129
A.3. Alfabeto con 37 símbolos . . . . .	130

<b>B. Máquinas de Turing</b>	<b>131</b>
------------------------------	------------

<b>C. Ejercicios propuestos en clase</b>	<b>135</b>
--	------------



# Índice de figuras

4.1. Suma de dos números en binario. . . . .	33
5.1. Dispositivo SBB. . . . .	42
5.2. Encriptación con DES. . . . .	42
12.1. Ejemplo de curva elíptica sobre $\mathbb{R}$ . . . . .	104
12.2. Ejemplo de la suma de dos elementos. . . . .	106
12.3. Ejemplo de la suma de un elemento consigo mismo. . . . .	107
12.4. Ejemplo de la suma de un elemento y su opuesto. . . . .	109
12.5. Comparación entre los métodos RSA y curvas elípticas. . . . .	121
B.1. Funcionamiento de una Máquina de Turing. . . . .	133



# Índice de cuadros

7.1. Tiempos de búsqueda de números primos. . . . .	61
A.1. Alfabeto de 29 símbolos. . . . .	128
A.2. Alfabeto de 36 símbolos. . . . .	129
A.3. Alfabeto de 37 símbolos. . . . .	130
B.1. Tabla de estados y símbolos de una Máquina de Turing. . . . .	132



# Parte I

## Introducción a la Criptografía



# Capítulo 1

## Introducción

La criptografía viene utilizándose desde la antigüedad, pero ha sido en los últimos años cuando ha empezado a ser conocida y utilizada por la gente de la “calle”.

### 1.1. ¿Qué es la criptografía?

Desde tiempos muy remotos el hombre ha tenido la necesidad de poder comunicarse de forma segura y confidencial. Para ello ha tenido que idear métodos y formas que le permitieran establecer comunicaciones de forma segura.

La criptografía es una de esas formas. Mediante la criptografía se podía transformar un mensaje de tal forma que el único capaz de entenderlo (descifrarlo) era la persona a la que iba dirigido.

Esto se hacía utilizando un algoritmo<sup>1</sup> y una clave. Un mensaje podía ser codificado de varias formas dependiendo de la clave utilizada.

El emisor y receptor del mensaje se ponían de acuerdo en la clave a utilizar, y es aquí donde estaba el peligro. Cualquiera que conociera la clave utilizada tendría acceso a las comunicaciones. El poder transmitir una clave de forma segura y que no fuera accesible por terceras partes ha sido uno de los grandes problemas de la criptografía que fue solucionado en los años 70 por “*Diffie*” y “*Hellman*”.

---

<sup>1</sup>Proceso que realiza la transformación.

La criptografía fue utilizada en la antigüedad de forma continua, ya que era necesario para las comunicaciones militares y diplomáticas. Julio Cesar cifraba todas sus comunicaciones. La criptografía no ha sido utilizada por los usuarios “normales” hasta los años 90 gracias a “Phill Zimmerman” y no ha tenido un uso masivo hasta la llegada de internet a los hogares ya que la cantidad de información que circula por la red es tan grande y muchas veces “sensible<sup>2</sup>” que es necesario protegerla de los “oidos indiscretos<sup>3</sup>” que pululan por la red.

## 1.2. Utilidades de la Criptografía

Mediante la criptografía podemos evitar que un documento digital que ha sido transmitido por un canal, sin protección alguna, sea modificado intencionadamente.

1. Podemos garantizar el origen de un documento, es decir, podemos verificar que el documento proviene de quien dice provenir.
2. Podemos garantizar la autenticidad del documento, es decir, que no ha sido modificado por ninguna persona.
3. Podemos verificar que el receptor es el receptor adecuado del documento, es decir, podemos evitar el entregar un documento a un impostor.

La criptografía nos ofrece seguridad en las comunicaciones digitales, siempre que se aplique de una forma correcta.

## 1.3. Criptografía Clásica

La criptografía es una forma de ocultar información, es decir una manera de transmitir información de forma confidencial. El hombre, desde los orígenes de su existencia, ha tenido la necesidad de poder transmitir mensajes de una forma segura y confidencial.

Podemos considerar las formas más primitivas<sup>4</sup> de lenguaje escrito como técnicas criptográficas, ya que eran muy pocos los capacitados para interpretar los símbolos utilizados. A pesar de esto era necesario enmascarar los

---

<sup>2</sup>Datos personales, económicos, ...

<sup>3</sup>La información que circula por internet pasa por muchos ordenadores antes de llegar a su destino, y si no está protegida puede ser leída por aquellos que tengan acceso a los ordenadores por los que pasa.

<sup>4</sup>Por ejemplo las escrituras jeroglíficas del antiguo Egipto.

símbolos que formaban los mensajes ya que aunque había pocas personas capacitadas para entender los mensajes no todas estaban autorizadas para hacerlo.

Una de las primeras formas de enmascarar información fue mediante una técnica conocida como *esteganografía*, la cual consistía en ocultar un mensaje secreto en otro mensaje inteligible. Por ejemplo:

- Hacer orificios sobre los caracteres que componían el mensaje secreto, por entre los cuales se hacían pasar filamentos constituyendo un trenzado que enmascaraba el mensaje original. Así pues, para leer dicho mensaje era necesario deshacer la maraña.
- Algunos espías en la Segunda Guerra Mundial escribían mensajes secretos con tinta invisible encima de otros mensajes en claro que enmascaraban a los primeros.

En la criptografía clásica los mensajes se transmitían cifrados con una clave secreta, de tal forma que el receptor pudiera descifrarlos utilizando la clave secreta con la que el emisor los cifró. Este tipo de criptografía también recibe el nombre de *criptografía simétrica de clase secreta*, puesto que emisor y receptor deben poseer la misma clave, secreta, para cifrar y descifrar los mensajes.

Las claves secretas utilizadas en la criptografía clásica debían ser transmitidas a través de canales seguros, normalmente por correos personales. Esto planteaba problemas:

1. Se debía disponer de correos fiables para el transporte de las claves.
2. En el caso de que el correo no pudiera llegar hasta el receptor, éste quedaba incomunicado.

## 1.4. Criptografía Moderna

El año 1,976 fue un año clave para la criptografía simétrica. En ese año un método de cifrado simétrico denominado *DES*<sup>5</sup> fue adoptado como estándar para la criptografía simétrica de clave secreta en los Estados Unidos y posteriormente en todo el mundo. El método fue desarrollado por IBM y refrendado

---

<sup>5</sup>Data Encryption Standard.

por la *NSA*<sup>6</sup> y el *NIST*<sup>7</sup>. Este procedimiento de cifrado es todavía hoy uno de los más utilizados, puesto que hasta el momento presente ha resistido todos los ataques conocidos y publicados en el mundo académico, si bien se ignora si sucede lo mismo con los ataques llevados a cabo por las distintas agencias de inteligencia gubernamentales. Sin embargo, aun en este nuevo procedimiento de cifrado, existía el problema del intercambio y distribución de claves secretas entre los comunicantes, actividades que debían llevarse a cabo a través de canales seguros. Este problema fue resuelto en ese mismo año por *W. Diffie* y *M. E. Hellman*<sup>8</sup> de la Universidad de Stanford en California mediante un protocolo criptográfico que permitía distribuir claves secretas a través de canales abiertos sin protección. Este invento fue el primer concepto innovador y revolucionario desde los tiempos de la criptografía clásica.

El procedimiento de intercambio de claves secretas de *Diffie* y *Hellman* estaba basado en funciones matemáticas cuyo cálculo directo es fácil pero el inverso tiene tal complejidad que o bien es imposible realizarlo con los conocimientos y ordenadores actuales o bien el tiempo necesario para realizarlo es tal que, cuando se logra descifrar la información con él cifrada dicha información no tiene utilidad. Con el método de *Diffie* y *Hellman* únicamente se podían intercambiar, pero predijeron que si se pudiesen encontrar funciones de una sola dirección con trampa, el conocimiento de dicha trampa permitiría el cálculo de la función inversa con la misma facilidad, y por tanto sería posible no sólo transmitir claves secretas a través de canales abiertos, sino también realizar operaciones de cifrado y descifrado de mensajes, así como producir firmas digitales.

---

<sup>6</sup>National Security Agency.

<sup>7</sup>Nacional Institute of Standards and Technology.

<sup>8</sup>Diffie, W., Hellman, M. E. "New Directions in Cryptography", IEEE Transactions on Information Theory, 1,976. IT-22, pp. 644-654.

En el año 1,977 *R. Rivest, A. Shamir y L. Adleman* del MIT<sup>9</sup> descubrieron una de dichas funciones y crearon un elegante método de cifrado y firma digital denominado *RSA*<sup>10</sup>. Esta aportación, junto con el protocolo de intercambio de claves de *Diffie y Hellman* fue el origen de la criptografía moderna o de clave pública.

En la criptografía de clave pública cada individuo posee dos claves, una secreta y otra pública, de forma que los cifrados realizados con una de las claves se descifran con la otra, siendo tanto el cifrado como el descifrado operaciones matemáticas realizables en tiempo real.

Las claves están relacionadas matemáticamente de tal forma que el conocimiento de la clave pública y la matemática que sustenta su relación con la secreta no es suficiente para el cálculo de ésta última. Para su cálculo se necesita conocer la “trampa”, que es secreta. En el caso de no conocer la “trampa” el cálculo de la clave secreta es tan complejo que no existe capacidad de cálculo suficiente para realizarlo a tiempo de utilizar dicha información con beneficio.

La “trampa” en el RSA es el conocimiento de la factorización de un número compuesto. Los números que se utilizan son del orden de 300 dígitos, cuya factorización requiere cientos de miles de años de CPU de superordenadores.

## 1.5. Ataques a la Seguridad Criptográfica

Como la criptografía surgió para esconder información a determinadas personas u organismos, luego siempre habrá alguien interesado en obtener una información cifrada. Por esta razón existen distintos métodos para obtener dicha información, los dos tipos básicos son:

- Ataques pasivos.

El intruso intenta obtener la información sin modificarla. Este tipo de ataque no puede ser detectado por los interesados, pero la criptografía ofrece protección contra este tipo de ataques, ya que evita que la información obtenida mediante la escucha sea inútil al estar cifrada.

---

<sup>9</sup>Massachusetts Institute of Technology.

<sup>10</sup>Rivest, R., Shamir, A., w,L. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems” *Communications of the ACM*, 21 (1,978), pp. 120-126.

- Ataques activos.

El intruso intenta obtener la información y modificarla a su conveniencia.

### **1.5.1. Ataque a partir del cifrado**

El atacante solamente tiene acceso al texto cifrado y a partir de él trata de encontrar la clave secreta con la que se cifró dicho texto.

Un método de cifrado que sea vulnerable a este tipo de ataque se considera absolutamente inseguro. La criptografía clásica fue víctima de estos ataques mediante el análisis estadístico de frecuencias de los símbolos en los distintos idiomas.

Los siguientes ataques son los más conocidos:

### **1.5.2. Ataque a partir del texto en claro**

El atacante tiene acceso a textos en claro y a sus correspondientes cifrados, a través de los cuales se intenta averiguar la clave secreta.

### **1.5.3. Ataque a partir del texto en claro elegido**

El atacante tiene la posibilidad de elegir un texto en claro y obtener su correspondiente cifrado.

### **1.5.4. Ataque a partir de texto en claro condicionado**

El atacante puede elegir un texto en claro condicionado por los cifrados previamente interceptados.

### **1.5.5. Ataque a partir de cifrado elegido**

El atacante elige un cifrado y obtiene el correspondiente texto en claro, es decir, el atacante tiene acceso al cifrador, pero no a la clave.

### **1.5.6. Ataque a partir de cifrado condicionado**

Es igual que el anterior, pero con la diferencia de que el cifrado elegido está condicionado por los textos en claro previamente obtenidos.

### **1.5.7. Ataque con clave conocida**

En este ataque el oponente obtiene algunas claves utilizadas en cifrados previos e intenta determinar nuevas claves.

### **1.5.8. Reutilización del protocolo**

En este caso el oponente realiza el ataque registrando una de las comunicaciones, o parte de ella, e intenta utilizarla insertándola en una comunicación posterior.

### **1.5.9. Suplantación de personalidad**

El atacante asume la identidad de uno de los comunicantes registrados en la red digital de comunicación.

### **1.5.10. Compilación de un diccionario**

Este ataque es típico de los sistemas que tienen un control de acceso. Cuando el usuario introduce su clave de acceso al sistema es transformada mediante una función y comparada con las transformadas en la base de datos del ordenador. Si la función de transformación es pública y el atacante tiene acceso a la línea de comunicación por la que se transmiten dichas transformadas, entonces puede almacenarlas en un registro. En estas condiciones, el ataque se lleva a cabo generando claves de acceso aleatorias y transformándolas hasta que alguna transformada coincida con otra de las previamente interceptadas.

### **1.5.11. Búsqueda exhaustiva**

Este ataque es similar al anterior y se lleva a cabo generando aleatoriamente todos los valores posibles de las claves de acceso y transformándolas. De esta forma se puede elegir la clave de acceso cuya transformada coincida con la interceptada.

### **1.5.12. Ataque mediante intromisión**

El atacante se introduce en la línea entre dos usuarios registrados, recibiendo la información de uno de ellos y enviándosela al otro después de modificarla.



# Capítulo 2

# Criptoanálisis



# Capítulo 3

## Sistemas Criptográficos

La idea básica de la criptografía ha permanecido constante a lo largo del tiempo, y básicamente es la siguiente:

Codificar un mensaje  $M$  en una cadena finita de símbolos, donde cada símbolo pertenece a un alfabeto finito denominado  $\Sigma$ .

### Definición 3.1 (Codificador)

Llamaremos codificador a una aplicación  $e$  tal que  $e(M) = C$ , donde  $M$  es el mensaje a codificar y  $C$  es el mensaje codificado. También es habitual que el codificador dependa de varios parámetros, por ejemplo  $e(M, K) = C$  donde  $K$  se denomina clave,  $C$  varía según el valor de  $K$ .

### Definición 3.2 (Decodificador)

Llamaremos decodificador a una aplicación  $d$  tal que  $d(C) = M$ , donde  $C$  es un texto codificado y  $M$  es el texto sin codificar. Al igual que en el codificador también es habitual utilizar una clave  $K$ , por ejemplo  $d(C, K) = M$ .

El texto  $M$  se suele denominar *texto plano* y  $C$  se suele denominar *cifrado*.

### Definición 3.3 (Sistema criptográfico)

Llamaremos sistema criptográfico a la terna  $(M, K, C)$  donde  $M$  y  $C$  son cadenas de alfabetos finitos  $\Sigma_1$  y  $\Sigma_2$  respectivamente y  $K$  es un conjunto finito de claves con la hipótesis de que existan las funciones  $e$  y  $d$ :

$$e : M \times K \longrightarrow C$$

$$d : C \times K \longrightarrow M$$

tales que  $d(e(M, K), K) = M$ .

**Definición 3.4 (Sistema criptográfico monoalfabético)**

*Un sistema criptográfico se dice monoalfabético si la codificación de cada símbolo es independiente del mensaje que se está codificando.*

**Definición 3.5 (Sistema criptográfico polialfabético)**

*Un sistema criptográfico se dice polialfabético si la codificación de algún símbolo depende del mensaje que se está codificando.*

El primer requisito para que un sistema criptográfico sea “seguro” es que sea polialfabético.

## 3.1. Algunos Sistemas Criptográficos

Veamos algunos de los sistemas criptográficos más simples:

### 3.1.1. Simple sustitución

Sea  $\Sigma$  el conjunto de los símbolos del lenguaje, por ejemplo el abecedario. Tomemos  $\pi$  como una permutación de  $\Sigma$ , es decir una aplicación biyectiva:

$$\pi : \Sigma \xrightarrow{\sim} \Sigma$$

obviamente  $\Sigma \neq Id$ .

Cifrar con este método sería sustituir cada palabra del mensaje por su imagen en la permutación  $\pi$ .

Este sistema es monoalfabético y es extremadamente vulnerable a ataques por análisis de frecuencias. Este sistema tiene el inconveniente de que si a lo largo del mensaje  $M$  se repite, por ejemplo, la cadena  $el$  varias veces y  $\pi(l) = r$  y  $\pi(e) = g$  entonces en el cifrado  $C$  se repetirá el mismo número de veces la cadena  $rg$ , lo cual nos da un patrón de comportamiento del algoritmo de cifrado.

### 3.1.2. Transposición de orden $d$

Dado un entero positivo  $d$  dividimos el mensaje  $M$  en bloques de longitud  $d$ . Entonces tomamos una permutación  $\pi$  de  $\{1, 2, \dots, d\}$  y aplicamos  $\pi$  a cada bloque. Por ejemplo para  $d = 5$  y  $\pi$  la siguiente permutación:

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 3 & 2 & 5 \end{pmatrix}$$

si tomamos el mensaje  $M = \text{Un curso de criptografía.}$  para codificarlo hemos de dividir el mensaje en bloques de longitud 5:

$$M = \text{Un cu|rso d|e cri|ptogr|afía.}$$

y en cada bloque aplicamos la permutación  $\pi$ , luego:

$$C = \text{cU nu| rosd|rec i|gpotr|aaíf.} = \text{cU nu rosdrec igpotraaíf.}$$

Para decodificar el mensaje unicamente tendremos que dividir  $C$  en bloques de longitud 5 y aplicar a cada bloque la permutación inversa  $\pi^{-1}$ :

$$\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 3 & 1 & 5 \end{pmatrix}$$

Como  $C = \text{cU nu| rosd|rec i|gpotr|aaíf.}$  entonces:

$$M = \text{Un cu|rso d|e cri|ptogr|afía.} = \text{Un curso de criptografía.}$$

Este método conserva la frecuencia de distribución de los símbolos, pero destruye los patrones de repetición que conservaba el método de *simple sustitución*.

### 3.1.3. Vigenere

En este sistema la clave consiste en una serie de  $d$  letras. Se asignan números a las letras del abecedario,  $A = 0, \dots, Z = 25$ . Se sitúa debajo del mensaje el conjunto de  $d$  letras de forma repetida y se suman, olvidando los espacios en blanco, modulo 26.

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

Para el caso  $d = 3$  y la secuencia  $AFG$  la clave será:

$$K = \text{AFGAFGAFGAFGAFGAFGAFG}$$

Luego para cifrar el mensaje:

$$\begin{aligned} M &= \text{UNCURSODECRIPTOGRAFIA} \\ K &= \text{AFGAFGAFGAFGAFGAFGAFG} \\ C &= \text{URIUXZUIKCX\tilde{N}PZUGXGFNG} \end{aligned}$$

### 3.1.4. El cifrador de César

Este sistema de cifrado lo utilizaba Julio César para enviar mensajes secretos. Es un sistema muy sencillo.

1. Asignamos a cada letra un número entre 0 y 25:

$$A = 0 \quad B = 1 \quad C = 2 \quad \dots \quad Z = 25$$

2. A cada letra del mensaje le sumamos  $3 \pmod{26}$ .

Para descifrar el mensaje lo unico que hay que hacer es restar 3 a cada letra del mensaje cifrado.

## 3.2. Sistemas de Cifrado

A la hora de cifrar un mensaje se puede hacer de dos formas:

- Cifrando símbolo a símbolo.
- Cifrando por bloques de símbolos.

### 3.2.1. Cifrado en flujo

En este tipo de cifrado el mensaje se cifra símbolo a símbolo. Por ejemplo el cifrado por simple sustitución<sup>1</sup> es un cifrado en flujo, ya que permuta cada símbolo, empezando por el primero y terminando por el último, de uno en uno.

### 3.2.2. Cifrado en bloque

En este tipo de cifrado se divide el mensaje en bloques de una determinada longitud  $d$  y se va cifrando bloque a bloque. Por ejemplo el cifrado por una transposición de orden  $d$ , en la página 24, es un cifrado en bloque, ya que divide el mensaje en bloques y cifra bloque a bloque.

---

<sup>1</sup>Apartado 3.1.1 en la página 24.

### 3.3. Transformaciones Lineales

La idea básica es dividir el mensaje en bloques de longitud  $d$  e identificar cada bloque con un vector  $x$  de enteros con dimensión  $d$  y multiplicar dicho vector por una matriz no singular<sup>2</sup>  $U$ . Este tipo de cifrado es un cifrado en bloque.

El cifrado se llevaría a cabo de la siguiente manera:

$$U \cdot x = y$$

donde  $y$  es el texto cifrado. Para descifrar el mensaje  $y$  habrá que multiplicarlo por  $U^{-1}$ , existe al ser  $U$  una matriz no singular:

$$U^{-1} \cdot y = x$$

Para asegurarnos que todas las operaciones aritmeticas se realizan sobre un cuerpo haremos los calculos modulo un número primo. Es decir, si hacemos los calculos modulo el número primo  $p$ , significa que estaremos trabajando en el cuerpo finito de  $p$  elementos  $\mathbb{F}_p$ .

Normalmente consideraremos dos alfabetos:

- Un alfabeto de 29 símbolos<sup>3</sup> donde:
  - Las letras A-Z están representadas por los enteros comprendidos entre 0 y 25, ambos inclusive.
  - El espacio en blanco está representado por 26.
  - La coma está representada por 27.
  - El punto y final está representado por 28.
- Un alfabeto alfa-numérico de 37 símbolos<sup>4</sup> donde:
  - Los enteros 0-9 están representados por ellos mismos.
  - El espacio en blanco está representado por 10.
  - Las letras A-Z están representadas por los enteros comprendidos entre 11 y 36, ambos inclusive.

---

<sup>2</sup>No singular en el cuerpo en el que estemos trabajando, el cual será un cuerpo finito.

<sup>3</sup>En la página 128.

<sup>4</sup>En la página 130.

**Ejemplo 3.1**

Utilizando el alfabeto de 37 letras y con una longitud de bloque  $d = 3$  queremos cifrar el mensaje “PRUEBA” utilizando como clave la matriz:

$$U = \begin{pmatrix} 5 & 0 & 3 \\ 11 & 7 & 8 \\ 4 & 23 & 36 \end{pmatrix}$$

El mensaje será:

$$M = PRUEBA = 27 \ 29 \ 32 \ 15 \ 12 \ 11$$

Luego para codificarlo:

$$\begin{pmatrix} 5 & 0 & 3 \\ 11 & 7 & 8 \\ 4 & 23 & 36 \end{pmatrix} \cdot \begin{pmatrix} 27 & 15 \\ 29 & 12 \\ 32 & 11 \end{pmatrix} = \begin{pmatrix} 231 & 108 \\ 756 & 337 \\ 1927 & 732 \end{pmatrix} \equiv \begin{pmatrix} 9 & 34 \\ 16 & 4 \\ 3 & 29 \end{pmatrix} \pmod{37}$$

de donde tenemos que el mensaje cifrado será:

$$C = 9 \ 16 \ 3 \ 34 \ 4 \ 29 = 9F3X4R$$

Para descifrar el mensaje tendremos que utilizar  $U^{-1}$ , es decir la inversa de  $U$ . Como en este caso estamos utilizando un alfabeto de 37 símbolos su inversa, en  $\mathbb{F}_{37}$ , será:

$$U^{-1} = \begin{pmatrix} 32 & 2 & 1 \\ 5 & 29 & 25 \\ 21 & 9 & 23 \end{pmatrix}$$

El mensaje descifrado:

$$\begin{pmatrix} 32 & 2 & 1 \\ 5 & 29 & 25 \\ 21 & 9 & 23 \end{pmatrix} \cdot \begin{pmatrix} 9 & 34 \\ 16 & 4 \\ 3 & 29 \end{pmatrix} = \begin{pmatrix} 323 & 1125 \\ 584 & 1011 \\ 402 & 1417 \end{pmatrix} \equiv \begin{pmatrix} 27 & 15 \\ 29 & 12 \\ 32 & 11 \end{pmatrix} \pmod{37}$$

$$C = 27 \ 29 \ 32 \ 15 \ 12 \ 11 = PRUEBA$$

**Observacion: 3.3.1**

En el ejemplo anterior hemos utilizado la matriz:

$$U = \begin{pmatrix} 5 & 0 & 3 \\ 11 & 7 & 8 \\ 4 & 23 & 36 \end{pmatrix}$$

cuyos coeficientes están en  $\mathbb{F}_{37}$ , ya que estamos utilizando un alfabeto de 37 símbolos. Su inversa en los números reales,  $\mathbb{R}$ , es:

$$\begin{pmatrix} \frac{68}{1015} & \frac{69}{1015} & \frac{-3}{145} \\ \frac{-52}{145} & \frac{24}{145} & \frac{-1}{145} \\ \frac{45}{203} & \frac{-23}{203} & \frac{1}{29} \end{pmatrix}$$

Para calcular esta matriz en  $\mathbb{F}_{37}$  tendremos que calcular los coeficientes en dicho cuerpo:

$$\left| \begin{array}{l} 68 \equiv 31 \pmod{37} \\ -52 \equiv 22 \pmod{37} \\ 45 \equiv 8 \pmod{37} \\ 1015 \equiv 16 \pmod{37} \\ 29 \equiv 29 \pmod{37} \end{array} \right| \left| \begin{array}{l} 69 \equiv 32 \pmod{37} \\ 24 \equiv 24 \pmod{37} \\ -23 \equiv 14 \pmod{37} \\ 145 \equiv 34 \pmod{37} \end{array} \right| \left| \begin{array}{l} -3 \equiv 34 \pmod{37} \\ -1 \equiv 36 \pmod{37} \\ 1 \equiv 1 \pmod{37} \\ 203 \equiv 18 \pmod{37} \end{array} \right|$$

Calculemos los inversos en  $\mathbb{F}_{37}$ :

$$\left| \begin{array}{l} 1015^{-1} \equiv 7 \text{ en } \mathbb{F}_{37} \\ 203^{-1} \equiv 35 \text{ en } \mathbb{F}_{37} \end{array} \right| \left| \begin{array}{l} 145^{-1} \equiv 12 \text{ en } \mathbb{F}_{37} \\ 29^{-1} \equiv 23 \text{ en } \mathbb{F}_{37} \end{array} \right|$$

Los coeficientes de  $U^{-1}$  en  $\mathbb{F}_{37}$  son:

$$\begin{aligned} \frac{68}{1015} &\equiv 31 \cdot 7 \text{ en } \mathbb{F}_{37} = 217 \equiv 32 \pmod{37} \\ \frac{69}{1015} &\equiv 32 \cdot 7 \text{ en } \mathbb{F}_{37} = 224 \equiv 2 \pmod{37} \\ \frac{-3}{145} &\equiv 34 \cdot 12 \text{ en } \mathbb{F}_{37} = 408 \equiv 1 \pmod{37} \\ \frac{-52}{145} &\equiv 22 \cdot 12 \text{ en } \mathbb{F}_{37} = 264 \equiv 5 \pmod{37} \\ \frac{24}{145} &\equiv 24 \cdot 12 \text{ en } \mathbb{F}_{37} = 288 \equiv 29 \pmod{37} \\ \frac{-1}{145} &\equiv 36 \cdot 12 \text{ en } \mathbb{F}_{37} = 432 \equiv 25 \pmod{37} \\ \frac{45}{203} &\equiv 8 \cdot 35 \text{ en } \mathbb{F}_{37} = 280 \equiv 21 \pmod{37} \\ \frac{-23}{203} &\equiv 14 \cdot 35 \text{ en } \mathbb{F}_{37} = 490 \equiv 9 \pmod{37} \\ \frac{1}{29} &\equiv 1 \cdot 23 \text{ en } \mathbb{F}_{37} = 23 \equiv 23 \pmod{37} \end{aligned}$$

Luego la matriz inversa de  $U$ ,  $U^{-1}$ , en  $\mathbb{F}_{37}$  es:

$$U^{-1} = \begin{pmatrix} 32 & 2 & 1 \\ 5 & 29 & 25 \\ 21 & 9 & 23 \end{pmatrix}$$

### 3.4. Transformaciones Afines

Este tipo de transformación es muy similar a la anterior, razón por la cual no la explicaremos con mucho detalle. Se trata de dividir el mensaje en bloque y cifrarlo con una aplicación del tipo:

$$T(u) = A \cdot u + B$$

donde  $A$  es una matriz, de orden  $m \times m$ , invertible en  $\mathbb{Z}/n$ ,  $n$  es la cantidad de símbolos que tiene nuestro alfabeto, y  $B$  un vector de longitud  $m$ . En este caso cifraremos el mensaje en bloques de  $m$  letras. Sean  $\{u_i\}$  el mensaje dividido en bloques de  $m$  letras, entonces para encriptar el mensaje haremos:

$$T(u_i) = A \cdot u_i + B = c_i$$

donde  $c_i$  es el el bloque  $u_i$  encriptado. Para desencriptar  $c_i$  bastará con despejar  $u_i$  de la ecuación anterior:

$$u_i = A^{-1} \cdot (c_i - B) = A^{-1} \cdot c_i - A^{-1} \cdot B$$

Podemos despejar ya que  $A$  es invertible en  $\mathbb{Z}/n$ .

# Capítulo 4

## Complejidad Computacional

Para cifrar mensajes necesitamos la existencia de una función  $e(M, K)$  la cual cifrará el mensaje  $M$  utilizando la clave  $K$ ,  $e(M, K) = C$ . Una vez cifrado el mensaje  $M$  es necesaria una función  $d(C, K)$  la cual descifrará el mensaje cifrado  $C$  con la clave  $K$ ,  $d(C, K) = M$ .

La existencia de la función inversa de la función  $e$ , es decir la función  $d$ , implica que cualquiera que conozca dicha función pueda descifrar los mensajes cifrados.

Para que un sistema criptográfico sea seguro será necesario que el calculo de la función  $d$  sea difícil, es decir, que tenga un alto coste computacional. Cuanto más difícil sea evaluar la función  $d$  más seguro será el sistema criptográfico ya que requerirá una cantidad mayor de tiempo el descifrar un mensaje cifrado.

Es lógico pensar que si evaluar  $d$  requiere de un alto coste computacional no sólo será un impedimento para aquellas personas ajenas al emisor y receptor del mensaje, sino que también lo será para el receptor del mensaje, y en principio así es. Este problema lo trataremos más adelante.

## 4.1. Como calcular el Coste Computacional

Podemos definir el coste computacional como el tiempo que se necesita para realizar alguna tarea en un ordenador. Obviamente este tiempo dependerá del ordenador que estemos utilizando y de las operaciones que necesita hacer el ordenador para realizar la tarea. Obviamente en un ordenador no se tarda el mismo tiempo en sumar dos enteros que en multiplicarlos, dividirlos o realizar una exponenciación.

El coste computacional no sólo depende del número de operaciones a realizar, sino que además depende el tiempo necesario para realizar cada una de ellas.

Entenderemos por coste computacional:

- A una función que depende del tamaño de la entrada, habitualmente denotada como  $n$ .
- Para un tamaño dado  $n$  entenderemos por coste computacional el peor de todos los casos posibles.

### 4.1.1. Operaciones a nivel de “bits”

Cualquier número con el que opere un ordenador estará en formato binario, es decir expresado en base dos. Por lo tanto para calcular el coste computacional de una operación calcularemos el número de operaciones que son necesarias, a nivel de “bits”, para resolverla.

“Bit” es la abreviatura, en inglés, de “binary digit”. Sea  $n \in \mathbb{Z}$  un número, entonces su expresión en binario será:

$$a_r a_{r-1} \dots a_1 a_0 = \sum_{i=0}^r a_i \cdot 2^i = n \quad \text{donde los } a_i \in \{0, 1\}$$

los  $a_i$  son los bits de  $n$  expresado en binario. Tenemos que:

$$r = \lceil \log_2 n \rceil + 1 = \left\lceil \frac{\log n}{\log 2} \right\rceil + 1$$

donde  $\lceil \log_2 n \rceil$  es el menor entero mayor que  $\log_2 n$ .

Supongamos que tenemos dos números  $a, b \in \mathbb{Z}$  y que ambos números tienen  $k$  bits de longitud<sup>1</sup>. Sean sus expresiones en binario las siguientes:

$$\begin{aligned} a &= a_k a_{k-1} \dots a_1 a_0 \\ b &= b_k b_{k-1} \dots b_1 b_0 \end{aligned}$$

vamos a determinar el número de operaciones necesarias para sumar ambos números, para ello veamos un ejemplo de la suma de dos números, en binario:

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ +\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 1\ 0 \end{array}$$

Figura 4.1: Suma de dos números en binario.

Luego para sumar dos números de  $k$  bits necesitaremos realizar  $2 \cdot k$  operaciones.

De igual forma se puede ver que multiplicar dos enteros de  $m$  y  $n$  bits necesita  $m \cdot n$  operaciones.

#### **Ejemplo 4.1 (Número de operaciones para calcular $n!$ )**

*Sea  $n$  un número entero cuya expresión en binario requiere  $k$  bits, a lo sumo. Como el número de bits del producto de dos enteros es menor o igual que la suma de las longitudes de dichos enteros tendremos entonces que el número de bits de  $n!$  será menor o igual que  $k \cdot n$ .*

*De la definición de  $n!$  se tiene que:*

$$n! = n \cdot (n-1) \cdot \dots \cdot 2$$

*luego será necesario realizar  $n-2$  productos para calcular  $n!$ .*

El procedimiento para calcular  $n!$  será multiplicar  $j!$  por  $j+1$ , donde  $j = 2, \dots, n-1$ .

---

<sup>1</sup>En el caso de no tener la misma longitud bastará con rellenar con ceros, por la izquierda el más pequeño de ellos.

Como  $2 \leq j \leq n - 1$  tendremos que  $j$  tendrá  $k$  bits a lo sumo, mientras que  $j!$  tendrá  $n \cdot k$  bits a lo sumo. Como multiplicar un número de  $k$  bits por otro de  $n \cdot k$  bits requiere  $n \cdot k^2$  operaciones tendremos que el cálculo de  $j! \cdot (j + 1)$  requiere  $n \cdot k^2$  operaciones, y dado que para calcular  $n!$  necesitamos hacer  $n - 2$  multiplicaciones del tipo  $j! \cdot (j + 1)$  tendremos que el número de operaciones será:

$$(n - 2) \cdot n \cdot k^2$$

Como  $n$  tiene  $k$  bits tendremos que:

$$k = \lceil \log_2 n \rceil + 1$$

Luego el número de operaciones necesarias para calcular  $n!$  será:

$$(n - 2) \cdot n \cdot (\lceil \log_2 n \rceil + 1)^2 \quad (4.1)$$

Podemos suponer que el número de operaciones es proporcional a  $n^2 \cdot \lceil \log 2n \rceil^2$  ya que si desarrollamos (4.1) vemos que el término que más aporta a la suma, a medida que crece  $n$ , es:

$$n^2 \cdot \lceil \log 2n \rceil^2$$

diremos entonces que el número de operaciones es del orden de  $n^2 \cdot \lceil \log 2n \rceil^2$  y lo denotaremos como:

$$O(n^2 \cdot \lceil \log 2n \rceil^2)$$

## 4.2. Tiempo Polinomial

Sea  $\mathcal{A}$  un algoritmo. Diremos que tiene una complejidad de tipo polinomial si existe algún polinomio  $p(x)$  tal que:

$$t_{\mathcal{A}}(n) \leq p(n)$$

donde  $t_{\mathcal{A}}(n)$  denota el máximo tiempo que necesita el algoritmo  $\mathcal{A}$  sobre todas las entradas de tamaño  $n$ .

Diremos que un problema se puede resolver en tiempo polinomial si existe algún algoritmo que lo resuelve y dicho algoritmo tiene una complejidad de tipo polinomial, en este caso diremos que es un problema de clase  $P$ .

Esta definición de tiempo polinomial no es muy rigurosa, daremos una definición más rigurosa utilizando máquinas de Turing.

### 4.2.1. Máquinas de Turing y el problema P

Utilizando las máquinas de Turing podemos dar una definición más rigurosa del problema P que la que hemos dado anteriormente.

Sea  $\Sigma$  un alfabeto y denotemos por  $\Sigma^*$  al conjunto de todas las cadenas finitas sobre el alfabeto  $\Sigma$ .

Sea  $M$  una máquina de Turing que para sea cual sea el elemento  $x \in \Sigma^*$  que utilizemos como entrada de dicha máquina. Entonces el coste computacional de  $M$  vendrá dado por la función:

$$t_M : \mathbb{Z}^+ \longrightarrow \mathbb{Z}^+$$

definida de la siguiente manera:

$$t_M(n) = \max_{|x|=n} \{t \text{ con } x \in \Sigma^*\}$$

donde  $t$  es el tiempo que tarda  $M$  en parar utilizando como entrada  $x$ , cadena finita de longitud  $n$ .

Diremos que una función  $f$  tiene una complejidad de tipo polinomial si  $f$  es computable Turing<sup>2</sup> y existe algún polinomio  $p(x)$  tal que  $t_M(n) \leq p(n)$  para todo  $n$ .

### 4.2.2. Problemas de tipo P

Todos aquellos problemas de tipo P son realizables, por un ordenador, en tiempo “razonable”. Con esto queremos decir que utilizando toda la potencia computacional disponible en el momento se puede realizar el algoritmo, aunque el tiempo que nos lleve para ello sea de varios meses o años.

Si el algoritmo para descifrar una clave criptográfica es de tiempo polinomial, es un problema de tipo P, dicho sistema no será seguro ya que es posible romperlo en un tiempo determinado.

Consideraremos que un sistema criptográfico es “seguro” si el algoritmo para romper dicho sistema es un algoritmo de tiempo exponencial, es decir, el tiempo necesario para romperlo aumenta de forma exponencial.

---

<sup>2</sup>Existe una máquina de Turing,  $M$ , que realiza  $f$ .



# Parte II

## Criptografía Clásica



# Capítulo 5

## Algoritmos Simétricos

Los algoritmos simétricos son algoritmos de cifrado en los que tanto el emisor como el receptor han de conocer las claves para cifrar y descifrar, es decir, tanto emisor como receptor utilizan la misma clave para cifrar y descifrar. Este método posee una desventaja, y es que se deben transmitir las claves por un canal seguro, algo muy difícil de asegurar.

Los algoritmos que hemos visto hasta aquí son todos algoritmos simétricos.

### 5.1. Sistemas de Encriptación con Estructura de Grupo

Una cosa a tener en cuenta en los sistemas de encriptación es si poseen o no estructura de grupo.

**Definición 5.1 (Estructura de grupo para sistemas de Encriptación)**

*Se dice que un sistema de encriptación posee estructura de grupo si se cumple que:*

$$\forall K_1, K_2 \quad \exists K_3 \text{ tal que } E_{K_2}(E_{K_1}(M)) = E_{K_3}(M)$$

*es decir, si hacemos dos cifrados encadenados con las claves  $K_1$  y  $K_2$ , existe una clave  $K_3$  que realiza la misma transformación.*

Es interesante que un algoritmo criptográfico, sea simétrico o no, no posea estructura de grupo, ya que si ciframos un mensaje con una clave  $K_1$  y luego ciframos el resultado con otra clave  $K_2$ , es como si hubiéramos empleado una clave de longitud doble, aumentando la seguridad del sistema. Si por el contrario el sistema criptográfico posee estructura de grupo y realizamos el mismo proceso no hubiéramos adelantado nada, ya que como posee estructura

de grupo lo que hemos hecho es equivalente a cifrar el mensaje con una cierta clave  $K_3$ , una sola vez.

## 5.2. El Algoritmo DES

El algoritmo DES es el algoritmo simétrico más utilizado, su nombre proviene de “Data Encryption Standard”. Este algoritmo fue presentado por IBM en el año 1.975 y fue adoptado como estándar federal en los Estados Unidos a principios de 1.977.

## 5.3. Descripción del DES

Este sistema cifra bloques de 64 bits con una clave de 64 bits, donde hay 8 bits de paridad, el último bit de cada byte es de paridad, con lo cual la clave es una clave de 56 bits.

Dado que la clave es de 56 bits tendremos que existen  $2^{56}$  posibles claves para este algoritmo.

De la clave inicial  $K$  se extraen 16 subclaves  $K_i$ , de 48 bits cada una.

Tenemos 16 “dispositivos”, a los que llamaremos SBB<sup>1</sup>, los cuales requieren como entrada un bloque de 64 bits y una subclave  $K_i$  de 48 bits, para producir como salida un bloque de 64 bits.

El algoritmo es el siguiente:

1. Se aplica una permutación inicial, IP, a cada bloque de 64 bits. Esto produce una salida  $B_j$ , donde  $B_j$  es el transformado del  $j$ -ésimo bloque de 64 bits por IP, de 64 bits.
2. Pasamos  $B_j$  por el primer SBB, la salida la pasamos por el segundo SBB y así sucesivamente con los 16 SBB. La clave que utiliza el  $i$ -ésimo SBB es la subclave  $K_i$ .
3. A la salida del último SBB le aplicamos la permutación  $IP^{-1}$ . Con lo cual obtenemos el texto cifrado.

---

<sup>1</sup>Standard Building Block, constructor estándar de bloques.

## 5.4. Encriptación con DES

El SBB es un “dispositivo” que toma como entrada dos bloques, uno de ellos de 64 bits y el otro de 48 bits.

Sea  $B = (b_{63}, \dots, b_0)$  el primer bloque y  $B' = (b'_{63}, \dots, b'_0)$  la salida producida por el SBB. Vamos a utilizar la siguiente notación:

$$\begin{aligned} L &= (b_{63}, \dots, b_{32}) \\ R &= (b_{31}, \dots, b_0) \\ L' &= (b'_{63}, \dots, b'_{32}) \\ R' &= (b'_{31}, \dots, b'_0) \end{aligned}$$

### 5.4.1. Procedimiento para obtener $L'$

El procedimiento para obtener  $L'$  es muy sencillo basta con tomar  $L' = R$ .

### 5.4.2. Procedimiento para obtener $R'$

El procedimiento para obtener  $R'$  es algo más complicado:

$$R' = L + f(K_i, R)$$

donde la “suma” es en  $\mathbb{F}_2$ . La función  $f(k_i, R)$  se define de la siguiente manera:

1. Se aplica una permutación de expansión, a la que llamaremos  $E$ , al bloque de 32 bits  $R$  para transformarlo en un bloque de 48 bits. Denotemos a esta operación como  $E(R)$
2. A continuación sumamos en binario  $E(R)$  con la subclave  $K_i$ :

$$E(R) + K_i$$

3. Aplicamos la permutación  $P$ , la cual transforma un bloque de 48 bits en otro de 32 a la suma anterior:

$$P(E(R) + K_i)$$

4. Sumamos, en binario, este resultado con  $R$ :

$$R + P(E(R) + K_i)$$

Luego tendremos que:

$$f(K_i, R) = R + P(E(R) + K_i)$$

de donde:

$$R' = L + f(K_i, R) = L + [R + P(E(R) + K_i)]$$

En las figuras 5.1 y 5.2 podemos ver un esquema del dispositivo SBB y del funcionamiento del DES.

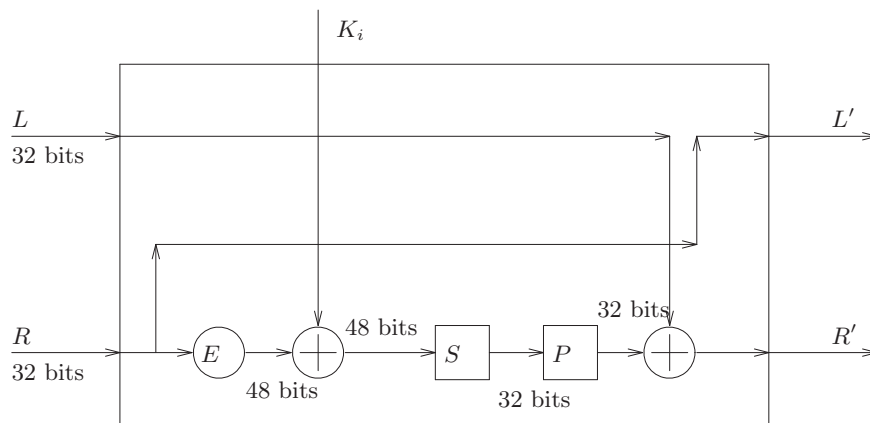


Figura 5.1: Dispositivo SBB.

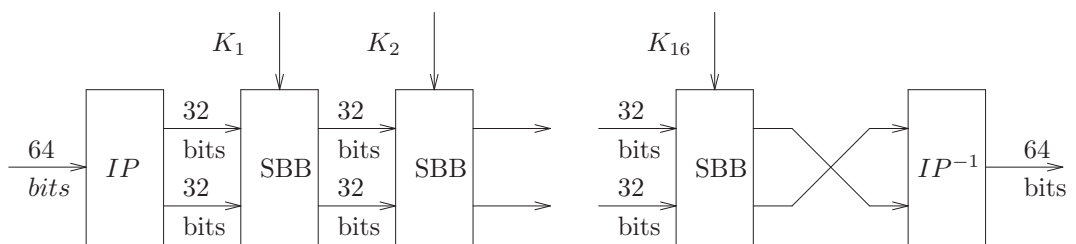


Figura 5.2: Encriptación con DES.

## 5.5. Descriptación con DES

Una vez tenemos un mensaje encriptado con DES el procedimiento para descifrarlo se realiza con el mismo algoritmo de encriptación, pero usando las llaves en orden inverso.

Supongamos que hemos encriptado un bloque de 64 bits con DES. Denotemos por:

$$\begin{aligned} L &= (b_{63}, \dots, b_{32}) \\ R &= (b_{31}, \dots, b_0) \end{aligned}$$

las partes izquierda y derecha del bloque antes de la encriptación, y por:

$$\begin{aligned} L' &= (b'_{63}, \dots, b'_{32}) \\ R' &= (b'_{31}, \dots, b'_0) \end{aligned}$$

las partes izquierda y derecha del bloque encriptado. Por el procedimiento de encriptación sabemos que:

$$\begin{aligned} L' &= R \\ R' &= L + f(K, R) \end{aligned}$$

despejando  $R$  y  $L$ , que son las partes que forman el mensaje original, tenemos que:

$$\begin{aligned} R &= L' \\ L &= R' - f(K, R) = R' - f(K, L') \end{aligned}$$

con lo que obtenemos los bloques del mensaje original.

## 5.6. Debilidad de las claves en el DES

Este algoritmo presenta algunas claves débiles. En general todos aquellos valores de la llave que conducen a una secuencia inadecuada de las subclaves  $K_i$  serán poco recomendables. Podemos distinguir los siguientes casos:

**Claves débiles** Son aquellas claves  $K$  que generan un conjunto de dieciséis subclaves  $K_i$  iguales entre si y que verifican:

$$E_K(E_K(M)) = M$$

es decir, encriptando un mensaje dos veces con la misma clave obtenemos el mensaje original. El procedimiento de encriptación, con esta clave, es una involución.

**Claves semidébiles** Son aquellas claves  $K$  que generan dos posibles valores para las dieciséis subclaves  $K_i$ , repitiéndose ocho veces cada posible valor.

El número de claves de este tipo es tan pequeño, en comparación con el número total de claves para el DES, que este problema no debería suponer motivo de preocupación.

## 5.7. Variantes del DES

A mediados de Julio de 1.998, una empresa sin ánimo de lucro, llamada EFF<sup>2</sup>, logró construir una máquina capaz de descifrar un mensaje encriptado con DES en menos de tres días. Curiosamente, pocas semanas antes, un alto cargo de la NSA había declarado que dicho algoritmo seguía siendo seguro, y que descifrar un mensaje resultaba aún excesivamente costoso, incluso para organizaciones gubernamentales. DES-Cracker costó menos de 40 millones de pesetas.

A pesar de este hecho, DES sigue siendo ampliamente utilizado en multitud de aplicaciones, como por ejemplo en las transacciones de los cajeros automáticos. De todas formas, el problema real de DES no radica en su diseño, sino en que emplea una clave demasiado corta, 56 bits, lo cual hace que con el avance actual de los ordenadores los ataques por fuerza bruta comiencen a ser las opciones más realistas. Mucha gente se resiste a abandonar este algoritmo, precisamente porque ha sido capaz de sobrevivir durante veinte años sin mostrar ninguna debilidad en su diseño, y prefieren proponer variantes que, de un lado evitarían el riesgo de tener que confiar en algoritmos nuevos, y de otro permitirían aprovechar gran parte de las implementaciones por hardware existentes de DES.

Algunas de las variantes del DES más utilizadas son las siguientes:

### 5.7.1. DES Múltiple

Consiste en aplicar varias veces el algoritmo DES, con diferentes claves, al mensaje original. Se puede hacer ya que DES no presenta estructura de grupo<sup>3</sup>. El más utilizado es el llamado Triple-DES:

$$C = E_{K_1}(D_{K_2}(E_{K_1}(M)))$$

El procedimiento es el siguiente:

1. Encriptamos con la clave  $K_1$ .
2. Desciframos con la clave  $K_2$ .
3. Volvemos a encriptar con la clave  $K_1$ .

La clave resultante es la concatenación de las claves  $K_1$  y  $K_2$ , con una longitud total de 112 bits.

---

<sup>2</sup>Electronic Frontier Foundation.

<sup>3</sup>Apartado 5.1 en la página 39.

### 5.7.2. DES con Subclaves Independientes

Consiste en emplear subclaves diferentes para cada una de las dieciséis rondas de DES. Puesto que estas subclaves son de 48 bits, la clave resultante tendría 768 bits en total.

Empleando una técnica conocida como “criptoanálisis diferencial” esta variante podía ser rota con  $2^{61}$  textos planos escogidos, por lo que en la práctica no presenta un avance sustancial sobre el DES estándar.

### 5.7.3. DES Generalizado

Esta variante emplea  $n$  trozos de 32 bits en cada ronda en lugar de dos, por lo que aumentamos tanto la longitud de la clave como el tamaño de mensaje que se puede codificar, manteniendo sin embargo el orden de complejidad del algoritmo. Se ha demostrado sin embargo que no sólo se gana poco en seguridad, sino que en muchos casos incluso se pierde.

## Parte III

# Breve introducción a la Teoría de Números



# Capítulo 6

## Los Números Primos

La criptografía moderna se apoya en los números primos.

### **Definición 6.1 (Números primos)**

*Sea  $a \in \mathbb{Z}$  diremos que es un número primo cuando no pueda ser dividido, de forma entera, por ningún otro número entero salvo por el mismo o por la unidad.*

Dados dos números cualesquiera diremos que son primos entre sí cuando su M.C.D. sea la unidad.

### **6.1. Infinitud de los Números Primos**

Las claves para el cifrado mediante métodos criptográficos se basan en la factorización de números primos. Si la cantidad de números primos fuera finita podría suponer un problema a la hora de elegir claves “seguras”. La cantidad de números primos es infinita, es decir, dado un número primo cualquiera siempre existirá otro número primo mayor que él, luego siempre existirán infinitos números primos mayores que uno dado.

**Teorema 6.1 (Euclides)**

*Hay infinitos números primos.*

**Demostración:**

Supongamos que existe un número finito de números primos:

$$p_1, p_2, \dots, p_n \tag{6.1}$$

Consideremos el número natural  $m = (p_1 \cdot p_2 \cdot \dots \cdot p_n) + 1$ . Por el teorema 6.3 admitirá una descomposición en números primos, luego será múltiplo de, al menos, un número primo. Este número primo será distinto a cualquiera de los números primos de (6.1) ya que al dividir  $m$  por  $p_i$  para  $i = 1, \dots, n$  el resto es uno. Luego existe, al menos, un número primo que no está en (6.1).

Si añadimos este número primo a la lista de números primos y repetimos el proceso volveremos a encontrar otro número primo que no habíamos considerado. Podemos repetir este proceso de forma infinita y siempre encontraremos un número primo que no hubiéramos considerado, lo que demuestra la infinitud de los números primos.

■

La demostración anterior está incluida en “*Los Elementos*” de “*Euclides*” y como se puede apreciar es una demostración simple e ingeniosa.

## 6.2. Distribución de los Números Primos

Hemos visto en la sección anterior que existen infinitos números primos, pero la forma en la que se distribuyen entre los números enteros es un misterio.

Un número es primo cuando no es divisible por ningún número, excepto por él y la unidad. Cuanto mayor es un número más condiciones tiene que verificar para ser primo, ya que no puede ser divisible por una mayor cantidad de números.

Para poder comprender la dificultad de establecer una regla en la distribución de los números primos veremos que dado un número natural cualquiera,  $n \in \mathbb{N}$ , siempre podremos encontrar  $n$  números consecutivos no primos.

Fijemos un número cualquiera  $n \in \mathbb{N}$  y a continuación consideremos todos los números enteros comprendidos entre  $(n + 1)! + 2$  y  $(n + 1)! + (n + 1)$ ,

todos estos números son compuestos, es decir, ninguno de ellos es primo. Cualquiera de estos números es de la forma  $(n+1)! + i$ , donde  $2 \leq i \leq n+1$ . Tendremos entonces que:

$$(n+1)! + i = (n+1) \cdot n \cdot (n-1) \cdot \dots \cdot 3 \cdot 2 \cdot 1 + i$$

y como  $0 \leq i \leq n+1$  tendremos que  $i$  divide a los dos sumandos de  $(n+1)! + i$  y como  $i$  es distinto de 1 y de  $(n+1)! + i$  tendremos que  $(n+1)! + i$  no es un número primo.

Con este método podremos encontrar tantos números compuestos consecutivos como queramos, por ejemplo, sabemos que los 100,000,000 de números enteros comprendidos entre  $100,000,001! + 2$  y  $100,000,001! + 100,000,001$  son compuestos.

Esto induce a pensar que después de  $100,000,001! + 100,000,001$  no existen números primos, pero como hemos visto en el apartado anterior existen infinitos números primos, luego después de dicho número podemos encontrar números primos.

La búsqueda de números primos grandes, en la que se basa la criptografía de clave pública, es un problema de difícil solución, en lo que a coste computacional se refiere, por eso en la práctica se utilizan algoritmos probabilísticos para la búsqueda de números primos. Este tipo de algoritmos no nos aseguran la primalidad de un determinado número, únicamente nos aseguran que un determinado número es primo con una probabilidad muy alta<sup>1</sup>.

El siguiente teorema nos da una idea de como están distribuidos los números primos.

**Teorema 6.2 (Teorema de los números primos)**

Sea la función  $\pi(x) =$  números primos menores o iguales que  $x$ . Tenemos que:

$$\pi(x) \simeq \frac{x}{\ln x}$$

cuando  $x \rightarrow \infty$ .

---

<sup>1</sup>Aunque la probabilidad es alta existe la posibilidad de que dicho número no sea primo.

### 6.3. Factorización de Números Enteros

#### Proposición 6.1

Fijados  $b, c \in \mathbb{Z}$  existen  $x_0, y_0 \in \mathbb{Z}$  tales que:

$$\text{M.C.D } (b, c) = (b, c) = b \cdot x_0 + c \cdot y_0$$

#### Demostración:

Consideremos el menor entero positivo,  $l$ , del siguiente conjunto:

$$\mathcal{C} = \{ b \cdot x + c \cdot y \text{ con } x, y \in \mathbb{Z} \}$$

luego  $l$  será de la forma  $l = b \cdot x_0 + c \cdot y_0$ .

Si  $l$  no divide a  $b$  entonces existiran  $q, r$  tales que:

$$b = l \cdot q + r \quad \text{con } 0 < r < l$$

por el algoritmo de Euclides para la división entera. Luego:

$$r = b - q \cdot l = b - q \cdot (b \cdot x_0 + c \cdot y_0) = b \cdot (1 - q \cdot x_0) + c \cdot (-q \cdot y_0)$$

de donde se deduce que  $r \in \mathcal{C}$  y como  $r$  es positivo y menor que  $l$  llegamos a contradicción, ya que  $l$  es el menor entero positivo de  $\mathcal{C}$ , luego  $l$  divide a  $b$ . Por el mismo motivo tendremos que  $l$  divide a  $c$ . Luego  $l$  dividirá a  $(b, c)$ .

Por definición de M.C.D. tendremos que  $(b, c)$  divide a  $b$  y a  $c$ , luego en particular dividirá a  $b \cdot x_0 + c \cdot y_0$ , lo que demuestra:

$$l = b \cdot x_0 + c \cdot y_0 = (b, c)$$

■

**Lema 6.1 (Lema de Euclides)**

*Si un número primo divide a un producto de números enteros, entonces divide a algún factor.*

**Demostración:**

Sea  $p \in \mathbb{N}$  un número primo tal que divide, de forma entera, a  $a \cdot n_1 \cdot \dots \cdot n_s$ , con  $a, n_1, \dots, n_s \in \mathbb{N}$ . Como  $n_1 \cdot \dots \cdot n_s \in \mathbb{N}$  llamando  $b = n_1 \cdot \dots \cdot n_s$  tendremos que  $p$  divide a  $a \cdot b$ , con lo cual reducimos el lema al caso en que un número primo divide al producto de dos enteros.

Si  $p$  no divide a  $a$  entonces  $(p, a) = 1$  y por la proposición 6.1 tendremos que existen  $x_0, y_0 \in \mathbb{Z}$  tales que:

$$1 = p \cdot x_0 + a \cdot y_0$$

multiplicando por  $b$  en esta igualdad tendremos:

$$b = (b \cdot p) \cdot x_0 + (b \cdot a) \cdot y_0$$

$p$  divide a  $(b \cdot p) \cdot x_0$ , y  $p$  también divide a  $(b \cdot a) \cdot y_0$  ya que por hipótesis:

$$a \cdot b = p \cdot n \quad n \in \mathbb{Z}$$

luego  $p$  divide a  $b$  ya que divide a los dos sumandos en los que descompone  $b$ .

■

**Teorema 6.3 (Teorema fundamental de la Aritmética)**

*Todo número natural mayor que la unidad es producto de números primos. Esta descomposición es única salvo el orden de los factores.*

**Demostración:**

La demostración la haremos por inducción. Para  $n = 2$  el teorema es cierto. Supongamos ahora el teorema cierto hasta  $m$  y veamos que, en efecto, el teorema se verifica para  $m + 1$ .

En el caso de que  $m + 1$  sea un número primo hemos terminado. Supongamos que  $m + 1$  no es un número primo, en ese caso  $m + 1$  tendrá dos divisores, al menos, distintos de 1 y  $m + 1$ .

$$m + 1 = a_1 \cdot a_2 \quad 1 < a_1, a_2 < m + 1 \quad a_1, a_2 \in \mathbb{N}$$

Por hipótesis de inducción tendremos que  $a_1$  y  $a_2$  descomponen en producto de números primos por ser menores que  $m + 1$ :

$$\begin{aligned} a_1 &= p_1^{n_1} \cdot \dots \cdot p_r^{n_r} \\ a_2 &= q_1^{l_1} \cdot \dots \cdot q_s^{l_s} \end{aligned}$$

con  $\{p_1, \dots, p_r, q_1, \dots, q_s\}$  números primos y  $\{n_1, \dots, n_r, l_1, \dots, l_s\} \in \mathbb{N}$ . Luego tendremos:

$$m + 1 = a_1 \cdot a_2 = p_1^{n_1} \cdot \dots \cdot p_r^{n_r} \cdot q_1^{l_1} \cdot \dots \cdot q_s^{l_s}$$

es decir  $m + 1$  descompone en producto de números primos.

Supongamos que un número admite dos descomposiciones, distintas, en producto de números primos:

$$\begin{aligned} n &= p_1^{n_1} \cdot \dots \cdot p_r^{n_r} \\ n &= q_1^{m_1} \cdot \dots \cdot q_s^{m_s} \end{aligned}$$

reordenando si hiciera falta podemos suponer que  $p_1 < p_2 < \dots < p_r$  y que  $q_1 < q_2 < \dots < q_s$ , en estos casos diremos que la descomposición es normal. Tendremos que  $p_i$  divide a  $q_1^{m_1} \cdot \dots \cdot q_s^{m_s}$  y por el Lema de Euclides  $p_i$  divide a alguno de los  $q_j$ , pero como los  $q_j$  son primos entonces  $p_i$  ha de ser uno de los  $q_j$ . De la misma forma cada  $q_j$  es uno de los  $p_i$ . Luego  $r = s$  y como las descomposiciones son normales entonces tendremos que  $p_i = q_i$  para todo  $i$ . Luego tenemos que:

$$\begin{aligned} n &= p_1^{n_1} \cdot \dots \cdot p_r^{n_r} \\ n &= p_1^{m_1} \cdot \dots \cdot p_r^{m_s} \end{aligned}$$

Supongamos que  $n_i > m_i$  entonces tendremos:

$$n = p_1^{n_1} \cdot \dots \cdot p_{i-1}^{n_{i-1}} \cdot p_i^{n_i - m_i} \cdot p_{i+1}^{n_{i+1}} \cdot \dots \cdot p_r^{n_r} = p_1^{m_1} \cdot \dots \cdot p_{i-1}^{m_{i-1}} \cdot p_{i+1}^{m_{i+1}} \cdot \dots \cdot p_r^{m_r}$$

luego  $p_i$  divide a  $p_1^{m_1} \cdot \dots \cdot p_{i-1}^{m_{i-1}} \cdot p_{i+1}^{m_{i+1}} \cdot \dots \cdot p_r^{m_r}$ , lo cual es imposible al ser los  $p_i$  números primos. Entonces ha de ser  $n_i \leq m_i$  y razonando de la misma manera obtendremos que  $n_i$  no puede ser menor que  $m_i$  luego la única posibilidad es que  $n_i = m_i$  para todo  $i$ .

■

## 6.4. Diferentes clases de Números Primos

Los números primos los podemos clasificar en varios tipos. Algunos de estos tipos son:

### 6.4.1. Números primos de Fermat

Estos números reciben su nombre en honor de “*Pierre de Fermat*”, que fue su descubridor. Estos números primos tienen la propiedad de ser de la forma:

$$p = 2^{2^n} + 1 \quad n \in \mathbb{N}$$

Los únicos números primos de Fermat conocidos son:

$$\begin{aligned} 3 &= 2^{2^0} + 1 & n &= 0 \\ 5 &= 2^{2^1} + 1 & n &= 1 \\ 17 &= 2^{2^2} + 1 & n &= 2 \\ 257 &= 2^{2^3} + 1 & n &= 3 \\ 65,537 &= 2^{2^4} + 1 & n &= 4 \end{aligned}$$

Como curiosidad sobre estos números podemos citar que el matemático “*Carl Friedrich Gauss*” caracterizó los polígonos regulares que podían ser construidos usando tan sólo regla y compás. Estos polígonos son aquellos que el número de sus lados es una potencia de dos o bien una potencia de dos multiplicada por uno o más primos de Fermat impares y distintos.

### 6.4.2. Números primos gemelos

Son números primos cuya diferencia es dos. Los siguientes pares son números primos gemelos:

$$\{3, 5\}, \{5, 7\}, \{11, 13\}, \{17, 19\}$$

### 6.4.3. Números primos de Mersenne

Los números de Mersenne son números enteros de la forma  $2^n - 1$ , con  $n$  un entero positivo. Si el número  $2^n - 1$  es un número primo entonces  $n$  es un número primo.

Luego llamaremos números primos de Mersenne a todos aquellos números primos que sean de la forma  $2^p - 1$ , donde  $p$  es un número primo, y a dichos números los denotaremos como  $M_p$ . En la actualidad no se sabe si existe o no una infinidad de números de Mersenne primos.

### 6.4.4. Números primos fuertes

La criptografía moderna se basa en la factorización de un número  $n$  en números primos. Normalmente  $n = p \cdot q$ , con  $p$  y  $q$  números primos. Para poder cifrar y descifrar es necesario conocer la factorización en números primos de  $n$ . Dado que  $n$  es un número grande dicha factorización es una tarea complicada. Podemos imponer una serie de condiciones adicionales a  $p$  y  $q$  para dificultar aún más la factorización de  $n$ .

Diremos que dos números primos  $p$  y  $q$  son primos fuertes si:

1. El máximo común divisor de  $p - 1$  y  $q - 1$  debe ser pequeño.
2.  $p - 1$  y  $q - 1$  deben tener algún factor primo grande  $p'$  y  $q'$ .
3. Tanto  $p' - 1$  como  $q' - 1$  deben tener factores primos grandes.
4. Tanto  $p' + 1$  como  $q' + 1$  deben tener factores primos grandes.

Las dos primeras condiciones se cumplen si tanto  $(p - 1)/2$  como  $(q - 1)/2$  son números primos.

# Capítulo 7

## Factorización y primalidad

La criptografía de clave pública se basa en la factorización de números primos. Para romper un sistema criptográfico de este tipo es necesario ser capaz de encontrar la descomposición de un número “grande” en factores primos. Los números utilizados en estos sistemas suelen ser producto de dos números primos de más de cien dígitos. Encontrar la factorización de números de este tipo es una tarea que requiere mucho tiempo. Es por este motivo que este tipo de sistemas criptográficos gozan de “gran” seguridad.

### 7.1. El Problema de la Factorización

El problema de factorizar un número entero en factores primos es un problema para el que no se han encontrado, todavía, algoritmos eficientes. La factorización de números requieren de bastante tiempo de cómputo.

Para factorizar un número en factores primos podemos dividir un número  $n$  por todos los primos anteriores a  $\sqrt{n}$ , pero para enteros de más de 15 cifras los resultados son insatisfactorios, desde el punto de vista del tiempo de cómputo.

En el año 1,975 un ingeniero inglés, “*John Pollard*” creó un método igual de simple, pero mucho más rápido que permite factorizar números de hasta 25 cifras sin demasiadas dificultades. Desgraciadamente este método es ineficiente cuando los números tienen más de 30 cifras. Investigaciones posteriores han dado origen a otros muchos algoritmos (método “rho” de Pollard, método  $p - 1$  de Pollard, criba cuadrática de Pomerance, método de las curvas elípticas de Lenstra, criba algebraica de Pollard<sup>1</sup>). Las técnicas son cada vez más eficaces y permiten factorizar cada vez números mayores, por supuesto también influye el progreso de los ordenadores.

Las técnicas de factorización no han alcanzado la eficacia que se tiene para los test de primalidad<sup>2</sup>. En el año 1,995 era posible factorizar un número de 90 cifras, utilizando sólo un potente microordenador, en unas pocas semanas. En el año 1,994, gracias a miles de ordenadores interconectados a través de internet, un grupo de matemáticos consiguió factorizar un número de 129 cifras.

Para los expertos en teoría de números la factorización de números “grandes” constituye un importante reto de tipo práctico.

## 7.2. Algoritmos

Los algoritmos que vamos a ver en este apartado son algoritmos muy básicos y no son utilizados en la práctica por ser algoritmos muy lentos, ya que en la práctica se utilizan números primos de más de 100 cifras y no es razonable intentar comprobar la primalidad de uno de estos números con los siguientes algoritmos. De todas formas estos algoritmos nos permitirán comprobar que la búsqueda de algoritmos para determinar la primalidad de números “grandes” no es una tarea sencilla.

---

<sup>1</sup>Este último es el más potente de todos.

<sup>2</sup>Pruebas para comprobar si un número es primo o no.

### 7.2.1. La criba de Eratóstenes

Eratóstenes fue un sabio Griego que vivió en el siglo III antes de nuestra era y dio un método para el cálculo de números primos. Dicho método se conoce como la criba de Eratóstenes.

La criba de Eratóstenes es un método elemental para encontrar los números primos que existen hasta un determinado valor entero. Por ejemplo, supongamos que queremos calcular todos los números primos comprendidos entre el 2 y el 100. Primero tacharemos todos los múltiplos de 2, luego tacharemos todos los múltiplos del siguiente número que no este tachado, a continuación tacharemos todos los múltiplos del siguiente número no tachado y así sucesivamente. Cuando hayamos terminado con todos los números entre el 2 y el 100 los números no tachados serán los números primos comprendidos entre el 1 y el 100.

Este método es útil para encontrar números primos pequeños, pero para números primos grandes es un método totalmente ineficiente.

### 7.2.2. Algoritmo de fuerza bruta

Un número es primo sí y sólo sí los únicos números que lo dividen son él mismo y la unidad. Luego dado un número si lo dividimos entre todos los números menores que él y distintos de la unidad y ninguno de ellos lo divide diremos que el número es primo.

Como todos los números pares son divisibles por dos tendremos que ningún número par, a excepción del dos, es primo. Luego podemos eliminar los números pares y buscar números primos entre los números impares.

Para calcular sí un número es primo o no con este método tendremos que dividirlo entre todos los números enteros, menores, positivos, no nulos y distintos de la unidad, comprobar los restos y si ningún resto es cero entonces dicho número es un número primo.

A este algoritmo nos referiremos como algoritmo I.

### 7.2.3. Algoritmo II

Si utilizamos los algoritmos anteriores para calcular números primos, por ejemplo los 10.000, 25.000, 100.000, ... primeros números primos veremos que a medida que aumentamos el número de primos que queremos calcular el tiempo de computo se dispara.

Para obtener un nuevo algoritmo de calculo, más rápido, podemos utilizar la descomposición de un número en números primos:

- Todo número entero descompone en producto de números primos. Para todo  $n \in \mathbb{N}$  se tiene:

$$n = p_1^{n_1} \cdot \dots \cdot p_r^{n_r}$$

donde  $n_i \in \mathbb{N}$  para  $i = 1, \dots, r$ ,  $p_j \in \mathbb{N}$  son primos para todo  $j$  y  $1 \leq p_j \leq n$  para  $j = 1, \dots, r$ . Esta descomposición es única salvo el orden.

Sea  $n \in \mathbb{N}$  un número cualquiera. Supongamos que queremos ver si es primo o no. En el caso de que no sea primo tendremos que existe  $m \in \mathbb{N}$  tal que  $1 < m < n$  y verificando  $n = m \cdot d$ .

Sea  $n = p_1^{m_1} \cdot \dots \cdot p_s^{m_s}$ , como  $m$  divide a  $n$  se tiene que cualquier número que divida a  $m$  dividirá a  $n$ , en particular los factores primos de  $m$   $\{p_1, \dots, p_s\}$ . Esto se deduce del hecho de que al dividir  $m$  a  $n$  tenemos que  $n$  es un múltiplo de  $m$ .

Este hecho nos permite simplificar los calculos para determinar sí un número no es primo, en lugar de dividir un número por todos los anteriores tendremos que dividirlo por unicamente por los primos anteriores a él.

Por ejemplo para determinar sí 100 es un número primo:

- Dividiendo por todos los anteriores a 100 tendríamos que dividir por 98 números.
- Dividiendo por los primos anteriores a 100 tendríamos que dividir por 25 números.

Realmente no tendríamos que dividir por todos esos números, ya que bastaría con encontrar un número que dividiera a 100, en este caso es facil 2, para determinar que no es primo.

Para determinar si un número es primo sí que tendríamos que dividir por todos los números anteriores, pero sí un número no es divisible, de forma entera, por todos los primos anteriores a él podemos afirmar con toda seguridad que es primo.

Por ejemplo para determinar que 287.117 es primo:

- Dividiendo por todos los anteriores a 287.117 tendríamos que dividir por 287.115 números.
- Dividiendo por todos los primos anteriores a 287.117 tendríamos que dividir por 24.999 números.

Como se puede ver con este método se reduce considerablemente el tiempo necesario para determinar sí un número es primo o no.

#### 7.2.4. Comparación de los algoritmos

Para comparar los diferentes algoritmos se han creado varios programas, en lenguaje C, que piden al usuario la cantidad de números primos. El programa buscará los 25.000, por ejemplo, primeros números primos. En todos los algoritmos se buscan números primos entre los números impares, a excepción del número dos.

Los calculos que muestra la tabla 7.1 se han realizado en un Pentium 166.

El 10.000 número primo es:	104.729
El 25.000 número primo es:	287.117
El 35.000 número primo es:	414.977
El 45.000 número primo es:	545.747
El 55.000 número primo es:	679.277

Cantidad	Algoritmo I	Algoritmo II
10,000	0 h 2 m 48 s	0 h 0 m 16 s
25,000	0 h 19 m 17 s	0 h 1 m 43 s
35,000	0 h 39 m 4 s	0 h 3 m 24 s
45,000	1 h 6 m 9 s	0 h 5 m 34 s
55,000	1 h 40 m 40 s	0 h 8 m 19 s

Cuadro 7.1: Tiempos de búsqueda de números primos.

### 7.3. Algoritmos Probabilísticos

Estos algoritmos son útiles para determinar si un número es primo o no cuando son números relativamente pequeños. Cuando los números a determinar si son primos o no son grandes estos algoritmos son totalmente ineficientes, aun utilizando grandes ordenadores.

#### 7.3.1. Segundo teorema de Fermat

El segundo teorema de Fermat es también conocido como “*Congruencia de Fermat*”, apartado (8.2.2) en la página 71.

Podemos enunciarlo de la siguiente manera:

Sean  $p, a \in \mathbb{Z}$  tales que  $p$  sea un número primo y no divida a  $a$  entonces  $a^{p-1} - 1$  es divisible por  $p$ .

Este teorema no impide la existencia de números no primos  $p$  que satisfagan la condición dada por el teorema, pero dichos números son raros.

Este método permite determinar cuando un número grande no es primo, y es más eficaz que los algoritmos dados en la sección (7.2) para números con cientos de cifras.

Comprobar si  $a^{p-1} - 1$  es divisible por  $p$  no es una tarea difícil, aun cuando  $p$  sea un número enorme.

Un método razonable para comprobar el carácter primo de un número impar  $n$  es el siguiente:

1. Comenzamos dividiendo el número  $n$  por los números impares 3, 5, 7, ... hasta un límite bastante pequeño. Si  $n$  no es divisible por ninguno de ellos se supone que el número es primo.
2. Ahora comprobamos la divisibilidad de  $2^{n-1} - 1$  por  $n$ . En el caso de que no sea divisible entonces el teorema de Fermat indica que  $n$  no es primo, ya que si lo fuera debería dividir a  $2^{n-1} - 1$  para satisfacer el segundo teorema de Fermat.

En el caso en que  $2^{n-1} - 1$  sea divisible por  $n$  no podemos asegurar que  $n$  sea un número primo, pero es muy probable que lo sea.

Este método es eficiente para determinar cuando un número no es primo, independientemente del número de cifras que tenga, pero tiene un inconveniente y es que no nos permite identificar números primos.

### 7.3.2. Algoritmos APRCL y Atkin-Morain

En el año 1,878 el matemático francés “*Edouart Lucas*” halló un test parecido al de Fermat, pero válido para los números de Mersenne<sup>3</sup>. Este test fue mejorado en el año 1.930 por el norteamericano “*Derrick H. Lehmer*”, es un algoritmo fácil de programar en un ordenador y permite demostrar el carácter primo de números gigantescos.

El test de “*Lucas-Lehmer*” se utiliza sistemáticamente para comprobar la fiabilidad de los superordenadores CRAY. En el año 1,994 el mayor número primo conocido fue descubierto por dos célebres cazadores de números primos “*David Slowinski*” y “*Paul Gage*”, en Estados Unidos. Dicho número es  $2^{859,433} - 1$  y tiene 258.716 cifras<sup>4</sup>, el ordenador utilizado para el cálculo de este número fue un CRAY C-90.

Pero este test posee el inconveniente de que sólo es aplicable a números de Mersenne. En el año 1.980 se logró desarrollar un algoritmo general capaz de demostrar el carácter primo de números “grandes”. La idea inicial partió de “*Len Adleman*”, de la universidad de Carolina del sur, “*Carl Pomerance*” y “*Robert Rumely*”, de la universidad de Georgia (Estados Unidos), y la versión práctica es debida a “*Hendrik Lensdra*”, de la universidad de Berkeley, y “*Henri Cohen*”. Dicho test se conoce como “APRCL”, las iniciales de sus inventores.

En 1.989 “*Oliver Atkin*”, de la universidad de Illinois en Estados Unidos, y “*François Morain*”, de la Escuela Politécnica de Palaiseau en Francia, crearon un algoritmo muy distinto de similar eficacia. La ventaja que tiene este método es la de poder comprobar, mediante los instrumentos de cálculo que suministra, la veracidad de que un número que el método afirma que es primo lo es realmente. En el caso del test APRCL, en cambio, el único medio de comprobación consiste en rehacerlo.

---

<sup>3</sup>Apartado (6.4.3) en la página 56.

<sup>4</sup>Razón por la que no indicamos “exactamente” dicho número. ;-)

Tanto el método APRCL como el método de Atkin-Morain emplean sofisticadas técnicas de teoría de números. El test APRCL utiliza una generalización del mencionado teorema de Fermat a los llamados *cuercos ciclotómicos*. El test de Atkin-Morain recurre a una generalización del mismo teorema a las curvas elípticas. En la práctica, estos dos modernos test permiten demostrar en pocos minutos que un número de 200 cifras es primo, lo que basta sobradamente para las necesidades criptográficas. Comparando este resultado con los obtenidos en la tabla 7.1 y teniendo en cuenta que el primo número 55.000 es 679.277 y tiene 6 cifras podemos hacernos una idea del tiempo que se tardaría en comprobar la primalidad de un número de 200 cifras con cualquiera de los métodos analizados en dicha tabla.

Podemos considerar que el problema de la determinación de la primalidad de un número es un problema asequible gracias a los algoritmos anteriores y al continuo avance de la tecnología que permite la utilización de ordenadores cada vez más rápidos y potentes.

### 7.3.3. Algoritmo de Lehmann

Es uno de los tests más sencillos para saber si un número  $p$  es primo o no. El algoritmo es el siguiente:

1. Escoger un número aleatorio  $a < p$ .

2. Calculamos:

$$b = a^{\frac{p-1}{2}} \pmod{p}$$

3. Si  $b \not\equiv 1 \pmod{p}$  y  $b \not\equiv p-1 \pmod{p}$  entonces  $p$  no es primo.

4. Si  $b \equiv 1 \pmod{p}$  o  $b \equiv p-1 \pmod{p}$ , la probabilidad de que  $p$  sea primo es igual o superior al 50 %.

Repetiendo el algoritmo  $n$  veces, la probabilidad de que  $p$  supere el test y no sea primo es de 1 contra  $2^n$ .

### 7.3.4. Algoritmo de Rabin-Miller

Es el algoritmo más empleado, debido a su facilidad de implementación. Sea  $p$  el número que queremos saber si es primo. Se calcula  $b$ , siendo  $b$  el número de veces que 2 divide a  $p - 1$ , es decir,  $2^b$  es la mayor potencia de 2 que divide a  $p - 1$ . Calculamos entonces  $m$ , tal que  $p = 1 + 2^b \cdot m$ .

1. Escoger un número aleatorio  $a < p$ .
2. Sea  $j = 0$  y  $z = a^m \bmod p$ .
3. Si  $z = 1$ , o  $z = p - 1$ , entonces  $p$  pasa el test y puede ser primo.
4. Si  $j > 0$  y  $z = 1$ ,  $p$  no es primo.
5. Sea  $j = j + 1$ . Si  $j = b$  y  $z \neq p - 1$ ,  $p$  no es primo.
6. Si  $j < b$  y  $z \neq p - 1$ ,  $z = z^2 \bmod p$ . Volver al paso (4).
7. Si  $j < b$  y  $z = p - 1$ , entonces  $p$  pasa el test y puede ser primo.
8.  $p$  no es primo.

La probabilidad de que un número compuesto pase este algoritmo para un número  $a$  es del 25%. Esto quiere decir que necesitaremos menos pasos para llegar al mismo nivel de confianza que el obtenido con el algoritmo de Lehmann.

### 7.3.5. En la práctica

Para determinar si un número es primo o no el algoritmo que normalmente se utiliza es:

1. Se genera un número aleatorio  $p$  de  $n$  bits.
2. Se pone a uno el bit más significativo para garantizar que el número obtenido es de  $n$  bits.
3. Se pone a uno el bit menos significativo para garantizar que el número es impar, ya que si fuera par no sería primo<sup>5</sup>.
4. Dividimos  $p$  por una tabla de primos precalculados, normalmente se utilizan primos menores que 2.000. Esto elimina gran cantidad de números no pbrimos de una forma rápida.
5. Ejecutar el test de Rabin-Miller como mínimo cinco veces.

---

<sup>5</sup>A excepción del 2, pero dicho número no se considera.



# Capítulo 8

## La Función $\Phi$ de Euler y Congruencias Clásicas

### 8.1. La Función $\Phi$ de Euler

Sea  $(\mathbb{Z}, +, \cdot)$  el anillo de los números enteros y  $n \in \mathbb{Z}$  un número cualquiera de dicho anillo. Podemos considerar entonces el anillo cociente de clases residuales resultante de dividir por  $n$ , al cual denotaremos por  $\mathbb{Z}/n$ .

$$\mathbb{Z}/n = \{ \overline{0}, \overline{1}, \dots, \overline{n-1} \}$$

El anillo  $\mathbb{Z}/n$  no siempre es cuerpo, es cuerpo cuando  $n$  es un número primo. Denotaremos por  $(\mathbb{Z}/n)^*$  al conjunto de todos los elementos de  $\mathbb{Z}/n$  que son invertibles, dicho conjunto es un grupo multiplicativo.

De la teoría de anillos sabemos que  $(\mathbb{Z}/n)^*$  estará formado por los números enteros, no nulos, menores que  $n$  y primos con  $n$ .

**Definición 8.1 (Función  $\Phi$  de Euler)**

*Se define la función  $\Phi$  de Euler como:*

$$\begin{aligned} \Phi : \mathbb{Z} &\longrightarrow \mathbb{Z} \\ n &\longrightarrow \Phi(n) \end{aligned}$$

donde  $\Phi(n) = |(\mathbb{Z}/n)^*|$ .

$\Phi(n)$  también recibe el nombre de indicador de Euler de  $n$ .

**Observacion: 8.1.1**

En el caso de que  $n = p$  sea un número primo se tiene que:

$$\Phi(p) = p - 1$$

ya que al ser  $p$  un número primo todos los números menores que  $p$  son primos con  $p$ .  $\mathbb{Z}/p$  es cuerpo por ser  $p$  primo, luego todos sus elementos, no nulos, son invertibles.

**8.1.1. Propiedades de la función  $\Phi$  de Euler****Proposición 8.1 (Propiedades del indicador de Euler)**

El indicador de Euler verifica:

- $\Phi(p^r) = p^{r-1} \cdot (p - 1)$  con  $p$  un número primo y  $r \geq 1$ .
- $\Phi(n \cdot m) = \Phi(n) \cdot \Phi(m)$  con  $n$  y  $m$  primos entre sí.

**Demostracion:**

- $\Phi(p^r) = p^{r-1} \cdot (p - 1)$  con  $p$  un número primo y  $r \geq 1$ .  
Por definición tendremos que  $\Phi(p^r)$  es el orden del grupo multiplicativo  $(\mathbb{Z}/p^r)^*$ , y este grupo está formado por los números enteros menores que  $p^r$  y que son primos con él. Los enteros menores que  $p^r$  y que no son primos con  $p^r$  son  $p, 2 \cdot p, \dots, p^{r-1} \cdot p$ . Luego tenemos  $p^{r-1}$  números que no son primos con  $p^r$  y como  $|\mathbb{Z}/p^r| = p^r$  tendremos que:

$$\Phi(p^r) = p^r - p^{r-1} = (p - 1) \cdot p^{r-1}$$

- $\Phi(n \cdot m) = \Phi(n) \cdot \Phi(m)$  con  $n$  y  $m$  primos entre sí.  
Por el Teorema Chino de los restos tenemos que:

$$\mathbb{Z}/(n \cdot m) \cong (\mathbb{Z}/n) \times (\mathbb{Z}/m)$$

induce un isomorfismo de grupos:

$$\mathbb{Z}/(n \cdot m)^* \cong (\mathbb{Z}/n)^* \times (\mathbb{Z}/m)^*$$

de donde se deduce que  $\Phi(n \cdot m) = \Phi(n) \cdot \Phi(m)$ .

■

## 8.2. Congruencias Clásicas

El concepto de congruencia fue introducido en Matemáticas por *Gauss* en su obra *Disquisitiones Arithmeticae*, en el año 1.801.

### Definición 8.2 (Números congruentes)

Sean  $a, b, m \in \mathbb{Z}^+ - \{0\}$ . Diremos que  $a$  y  $b$  son congruentes módulo  $m$  si  $m$  divide a  $(a - b)$  y lo denotaremos como  $a \equiv b \pmod{m}$ .

### Proposición 8.2

Sean  $a, b, m \in \mathbb{Z}^+ - \{0\}$ .

$a \equiv b \pmod{m} \iff a$  y  $b$  tienen el mismo resto al dividir por  $m$ .

### Demostración:

$\Rightarrow$  | Como  $a \equiv b \pmod{m}$  se tiene que  $m$  divide a  $(a - b)$  entonces tendremos por el algoritmo de Euclides que:

$$(a - b) = m \cdot \lambda \tag{8.1}$$

Por el algoritmo de Euclides tendremos:

$$\begin{aligned} a &= m \cdot \alpha_a + r_a \\ b &= m \cdot \alpha_b + r_b \end{aligned}$$

de donde se tiene que:

$$(a - b) = (\alpha_a - \alpha_b) \cdot m + (r_a - r_b) \tag{8.2}$$

De (8.1) y (8.2) se deduce que:

$$r_a - r_b = 0 \implies r_a = r_b$$

$\Leftarrow$  | Sean  $a, b \in \mathbb{Z}$  tales que tienen el mismo resto al dividir por  $m$ , entonces:

$$\begin{aligned} a &= m \cdot \alpha_a + r \\ b &= m \cdot \alpha_b + r \end{aligned}$$

de donde se tiene que:

$$a - b = m \cdot \alpha_a + r - (m \cdot \alpha_b + r) = (\alpha_a - \alpha_b) \cdot m$$

Luego  $m$  divide a  $(a - b)$ , es decir  $a \equiv b \pmod{m}$ .

■

De esta proposición se deduce que dos números son congruentes módulo  $m$  sí y sólo sí tienen el mismo resto al dividir por  $m$ .

Sí  $a \equiv b \pmod{m}$  y  $c \equiv d \pmod{m}$  se verifican las siguientes propiedades:

1.  $a + c \equiv b + d \pmod{m}$ .
2.  $k \cdot a \equiv k \cdot b \pmod{m}$  para todo  $k \in \mathbb{Z}$ .
3.  $a \cdot c \equiv b \cdot d \pmod{m}$ .
4.  $a^n \equiv b^n \pmod{m}$  para todo  $n \in \mathbb{Z}^+$ .
5.  $f(a) \equiv f(b) \pmod{m}$  para todo polinomio con coeficientes enteros.

### 8.2.1. Congruencia de Euler

#### Proposición 8.3 (Congruencia de Euler)

*Sí  $a, b \in \mathbb{Z}$  son primos entre sí entonces:*

$$a^{\Phi(b)} \equiv 1 \pmod{b}$$

#### Demostración:

Como  $a$  es primo con  $m$  entonces la clase de  $a$  en  $\mathbb{Z}/b$  es un elemento invertible:

$$\bar{a} \in (\mathbb{Z}/b)^*$$

y al ser  $(\mathbb{Z}/b)^*$  un grupo multiplicativo de orden  $\Phi(b)$  se tiene que:

$$1 = \bar{a}^{\Phi(b)} = a^{\Phi(b)}$$

■

### 8.2.2. Congruencia de Fermat

Un caso particular de la congruencia de Euler es la congruencia de Fermat:

**Proposición 8.4 (Congruencia de Fermat)**

*Sea  $p \in \mathbb{Z}$  un número primo y  $a \in \mathbb{Z}$  tal que no sea divisible por  $p$  entonces:*

$$a^{p-1} \equiv 1 \pmod{p}$$

**Demostración:**

Como  $p$  es primo y no divide a  $a$  entonces  $a$  y  $p$  son primos entre sí y por la congruencia de Euler tendremos:

$$a^{\Phi(p)} \equiv 1 \pmod{p}$$

pero como  $p$  es primo entonces  $\Phi(p) = p - 1$  luego:

$$a^{p-1} \equiv 1 \pmod{p}$$

■



# Capítulo 9

## El Algoritmo de Euclides

### Definición 9.1 (Anillo íntegro)

Diremos que un anillo  $A$  es íntegro cuando para cualesquiera  $a, b \in A$  tales que  $a \cdot b = 0$  se tenga que:

$$a = 0 \quad \text{o} \quad b = 0$$

### Definición 9.2 (Anillo Euclídeo)

Diremos que un anillo  $A$  es euclídeo si es íntegro y se verifica el algoritmo de Euclides para todo elemento del anillo.

Sea  $A$  un anillo Euclídeo y  $a, d \in A$  entonces:

$$a = d \cdot c + r \tag{9.1}$$

donde:

**a** recibe el nombre de dividendo.

**d** recibe el nombre de divisor.

**c** recibe el nombre de cociente.

**r** recibe el nombre de resto.

Donde (9.1) recibe el nombre de Algoritmo de Euclides. Si  $A = \mathbb{Z}$  diremos que (9.1) es el Algoritmo de Euclides para la división entera.

Ejemplo de anillos Euclídeos son  $\mathbb{Z}$  y  $\mathbb{K}[x]$ , donde  $\mathbb{K}[x]$  es el anillo de polinomios, en la variable  $x$ , con coeficientes en el cuerpo  $\mathbb{K}$ .

## 9.1. Cálculo del M.C.D. mediante el Algoritmo de Euclides

### Proposición 9.1

Sean  $a, d, c, r \in \mathbb{Z}$  tales que  $a = d \cdot c + r$  entonces:

$$M.C.D.(a, d) = M.C.D.(d, r)$$

### Demostración:

Por definición de M.C.D. tendremos que:

$$\begin{aligned} m_1 = M.C.D.(a, d) &\iff (m_1) = (a) + (d) \\ m_2 = M.C.D.(d, r) &\iff (m_2) = (d) + (r) \end{aligned}$$

$m_1$  y  $m_2$  existen y son únicos. Para demostrar la proposición veamos que  $(m_1) = (m_2)$ .

Esto es inmediato ya que:

$$\begin{aligned} a &= d \cdot c + r \in (d) + (r) \\ r &= a - d \cdot c \in (a) + (d) \end{aligned}$$

luego  $(a) + (d) = (d) + (r)$  luego  $m_1 = m_2$ .

■

Utilizando la proposición 9.1 podemos obtener un algoritmo rápido para determinar el M.C.D. de dos números, ya que la factorización en números primos de un número puede ser una tarea lenta y costosa, sobre todo si el número es grande.



### 9.2.1. Caracterización de los invertibles de $\mathbb{Z}/n$

#### Proposición 9.2

$\bar{m} \in \mathbb{Z}/n$  es invertible  $\iff (m, n) = 1$ ,  $m$  y  $n$  son primos entre sí.

#### Demostración:

$\Rightarrow$  |  $\bar{m}$  es invertible, entonces existe  $\bar{a} \in \mathbb{Z}/n$  tal que:

$$a \cdot m \equiv 1 \pmod{n} \implies a \cdot m - 1 \equiv 0 \pmod{n}$$

de donde se deduce que:

$$a \cdot m - \lambda \cdot n = 1 \text{ con } \lambda \in \mathbb{Z}$$

y por el ejercicio 1, en la página 135, se tiene que  $1 = (m, n)$ .

$\Leftarrow$  | Como  $(m, n) = 1$  por el ejercicio 1, en la página 135, tendremos que existen  $\lambda, \mu \in \mathbb{Z}$  tales que:

$$\lambda \cdot m + \mu \cdot n = 1$$

de donde se deduce:

$$\lambda \cdot m - 1 = -\mu \cdot n \iff \lambda \cdot m \equiv 1 \pmod{n}$$

luego  $\bar{\lambda} \in \mathbb{Z}/n$  es el inverso de  $\bar{m}$ .

■

Al conjunto de elementos invertibles de  $\mathbb{Z}/n$  lo denotaremos por  $(\mathbb{Z}/n)^*$  y estará formado por todos los elementos que son primos con  $n$ . Este conjunto es un grupo multiplicativo de orden  $\Phi(n)$ .

### 9.2.2. Ecuaciones diofánticas en $\mathbb{Z}$

#### Definición 9.3 (Ecuación diofántica en $\mathbb{Z}$ )

Dados  $a, b, c \in \mathbb{Z}$  llamaremos ecuación diofántica a la siguiente ecuación:

$$a \cdot x + b \cdot y = c$$

Diremos que tiene solución cuando existan  $x, y \in \mathbb{Z}$  que la verifiquen.

El ejercicio 1, en la página 135, nos indica cuando una ecuación diofántica tiene solución.

Una ecuación diofántica, como la anterior, tiene solución  $\iff c$  es múltiplo de máximo común divisor de  $a$  y  $b$ .

Sea  $(a, b) = d$  y  $c = \bar{c} \cdot d$ , tendremos que  $a = \bar{a} \cdot d$  y  $b = \bar{b} \cdot d$  entonces la siguientes ecuación diofánticas tendrán las mismas soluciones:

$$a \cdot x + b \cdot y = c \tag{9.2}$$

$$\bar{a} \cdot x + \bar{b} \cdot y = \bar{c} \tag{9.3}$$

Dada una ecuación diofántica cualquiera, que tenga solución, como la (9.2) podemos reducirla a otra ecuación diofántica que tiene las mismas soluciones y verificando  $(\bar{a}, \bar{b}) = 1$ , ecuación (9.3).

Cuando tengamos una ecuación diofántica del tipo (9.2) la reduciremos al tipo (9.3) calculando el máximo común divisor de  $a$  y  $b$ , mediante el algoritmo de Euclides por ejemplo, y tendremos que:

$$\bar{a} = \frac{a}{d} \quad \bar{b} = \frac{b}{d} \quad \bar{c} = \frac{c}{d}$$

Supondremos de ahora en adelante que si  $a \cdot x + b \cdot y = c$  es una ecuación diofántica entonces  $a$  y  $b$  son primos entre sí, y diremos que la ecuación esta en forma reducida.

Sea  $a \cdot x + b \cdot y = c$  una ecuación diofántica en forma reducida, si  $x_0, y_0 \in \mathbb{Z}$  son solución de dicha ecuación entonces  $(x_0 + \lambda \cdot b, y_0 - \lambda \cdot a)$  son soluciones de la ecuación para todo  $\lambda \in \mathbb{Z}$ .

**Ejemplo 9.1 (Resolución de una ecuación diofántica)**

Resolver la ecuación diofántica siguiente:

$$238 \cdot x + 165 \cdot y = 1$$

Lo primero que tenemos que ver es si tiene solución. Calculemos el máximo común divisor por el algoritmo de Euclides:

$$\begin{aligned} 238 &= 165 \cdot 1 + 73 &\implies 73 &= 238 - 165 \cdot 1 \\ 165 &= 73 \cdot 2 + 19 &\implies 19 &= 165 - 73 \cdot 2 \\ 73 &= 19 \cdot 3 + 16 &\implies 16 &= 73 - 19 \cdot 3 \\ 19 &= 16 \cdot 1 + 3 &\implies 3 &= 19 - 16 \cdot 1 \\ 16 &= 3 \cdot 5 + 1 \\ 3 &= 1 \cdot 3 \end{aligned}$$

luego  $(238, 165) = 1$  entonces son primos entre sí, la ecuación tiene solución. Para encontrar una solución procederemos de la siguiente manera:

$$\begin{aligned} 16 &= \underbrace{(19 - 16 \cdot 1)}_3 \cdot 5 + 1 &\implies 16 \cdot 6 + 19 \cdot (-5) &= 1 \\ 1 &= \underbrace{(73 - 19 \cdot 3)}_{16} \cdot 6 + 19 \cdot (-5) &\implies 73 \cdot 6 + 19 \cdot (-23) &= 1 \\ 1 &= \underbrace{(165 - 73 \cdot 2)}_{16} \cdot (-23) + 73 \cdot 6 &\implies 73 \cdot 52 + 165 \cdot (-23) &= 1 \\ 1 &= \underbrace{(238 - 165 \cdot 1)}_{73} \cdot 52 + 165 \cdot (-23) &\implies 238 \cdot 52 + 165 \cdot (-75) &= 1 \end{aligned}$$

Luego las soluciones de la ecuación diofántica son  $x_0 = 52$  e  $y_0 = -75$ ,  $(52, -75)$ . Todas las soluciones de la ecuación son:

$$(52 + \lambda \cdot 165, -75 - \lambda \cdot 238) \text{ con } \lambda \in \mathbb{Z}$$

**9.2.3. Cálculo de inversos con el algoritmo de Euclides**

Sea  $n, m \in \mathbb{Z}$ , con  $n$  positivo, no nulo y no primo, y que queremos calcular el inverso de  $m$  en  $\mathbb{Z}/n$ , que no es cuerpo ya que  $n$  no es primo. Podemos calcularlo utilizando el algoritmo de Euclides:

1. Comprobaremos que, efectivamente,  $m$  es invertible en  $\mathbb{Z}/n$ . Es decir que  $(m, n) = 1$ . Una forma de hacerlo es la descrita en el apartado (9.1), en la página 74.
2. En el caso en que  $m$  sea invertible tendremos que existirá  $x \in \mathbb{Z}$  tal que:

$$\bar{x} \cdot \bar{m} = \bar{1} \pmod{n}$$

Que un número sea congruente con 1 modulo  $n$  significa que:

$$x \cdot m = b \cdot n + 1 \text{ con } b \in \mathbb{Z}$$

Luego tenemos la siguiente ecuación diofántica:

$$x \cdot m + b \cdot n = 1$$

Resolvemos la ecuación diofántica como hemos visto en el apartado (9.2.2), en la página 77, y  $m^{-1} = x \pmod{n}$ .



**Parte IV**  
**Criptografía Moderna**



# Capítulo 10

## Funciones de un Sentido

Habíamos visto en el capítulo 4 que para que un sistema criptográfico sea seguro era necesario que la función  $d(\cdot, \cdot)$  requiera mucho tiempo de cálculo, con lo cual sería difícil descifrar un mensaje. Pero esto representa un problema, y es que no sólo será difícil para un intruso, además también lo será para el receptor del mensaje.

Una solución a esto la dieron *Diffie* y *Hellman*, consistía en utilizar funciones matemáticas tales que el cálculo de la función directa<sup>1</sup> sea fácil, pero el cálculo de la función inversa<sup>2</sup> sea difícil, pero que dicha función inversa tenga una “trampa”, y el conocimiento de dicha trampa permita un sencillo cálculo de la función inversa. Este tipo de funciones reciben el nombre de funciones de un sentido o funciones con trampa.

De esta definición podemos observar que el hecho de que una función sea de un sentido depende del tiempo de cálculo utilizado por un ordenador para su función inversa, es decir que depende del ordenador utilizado, ya que no es lo mismo realizar una operación en un ordenador personal que en una estación de trabajo o un superordenador, como pudiera ser un CRAY. De esto también se deduce que una función que hoy en día es considerada como de un sentido dentro de unos años no lo sea debido a que la tecnología ha avanzado lo suficiente como para fabricar ordenadores con los que sea posible calcular la inversa de la función en un tiempo razonable, en dicho momento la función dejaría de ser una función de un sentido.

---

<sup>1</sup>La función que cifra el mensaje.

<sup>2</sup>La que descifra el mensaje.

## 10.1. El Logaritmo Discreto

El logaritmo discreto es la función más comúnmente usada como función de un sentido.

Por la proposición 9.2, en la página 76, tenemos que  $(\mathbb{Z}/n)^*$  con  $n \in \mathbb{N}$  está formado por los enteros menores que  $n$  y primos con él. Además tenemos que  $|(\mathbb{Z}/n)^*| = \Phi(n)$ .

Por la congruencia de Fermat<sup>3</sup> tenemos que:

$$a^{\Phi(p)} = a^{p-1} \equiv 1 \pmod{p}$$

donde  $p$  es primo y  $a \in \mathbb{Z}/p$ .

Sea  $n \in \mathbb{N}$ , un entero positivo, y  $a$  un cualquier entero tal que  $1 \leq a \leq n$  verificando que  $a \in (\mathbb{Z}/n)^*$  y verificando la siguiente propiedad:

$$a^d \neq 1 \pmod{n} \quad \text{con } 1 \leq d < \Phi(n)$$

entonces diremos que  $a$  es una raíz primitiva de  $n$ . No todo entero tiene raíces primitivas. Aquellos enteros que tienen raíces primitivas están caracterizados por el siguiente teorema elemental de la teoría de números:

### Teorema 10.1

1. El entero  $n$  tiene una raíz primitiva si y solo si  $n$  es 1, 2, o 4, o si es de la forma  $p^k$  o  $2 \cdot p^k$ , donde  $p$  es un primo impar.
2. Si  $n$  tiene una raíz primitiva entonces tiene  $\Phi(\Phi(n))$  raíces primitivas.

Preguntas como la forma en la que las raíces primitivas están distribuidas o que enteros son raíces primitivas de algún primo son preguntas a las que todavía no se ha encontrado respuesta. Existe una famosa conjetura de Artin(1,927):

Cada entero que no es un cuadrado perfecto es raíz primitiva de algún primo.

Dado un entero  $n$  no se conoce ningún algoritmo para calcular una raíz primitiva suya en tiempo polinomial.

---

<sup>3</sup>En el apartado 8.2.2 en la página 71.

Teniendo en cuenta todo lo visto hasta aquí vamos a definir el logaritmo discreto.

### 10.1.1. El problema del Logaritmo Discreto

Sea  $n \in \mathbb{Z}$  un entero cualquiera que tenga una raíz primitiva, a la que denotaremos  $a$ . Para cualquier  $x$  tal que  $0 \leq x < \Phi(n)$  si:

$$y = a^x \pmod{n} \quad (10.1)$$

entonces diremos que  $x$  es el *logaritmo discreto* de  $y$  en la base  $a$  módulo  $n$  y lo denotaremos como  $x = \log_a y \pmod{n}$ .

La ecuación 10.1 tiene solución única y ello es debido a que  $a$  es una raíz primitiva.

#### **Teorema 10.2 (El logaritmo discreto está bien definido)**

*Sea  $n \in \mathbb{Z}$  un entero cualquiera que tenga una raíz primitiva, a la que denotaremos  $a$ . Existe un único  $x$  tal que  $0 \leq x < \Phi(n)$  verificando:*

$$y = a^x \pmod{n}$$

#### **Demostración:**

Como  $a$  es raíz primitiva de  $n$  tendremos:

$$a \in (\mathbb{Z}/n)^*$$

con

$$a^d \neq 1 \pmod{n} \quad \text{para cualquier } d \text{ tal que } 1 \leq d < \Phi(n)$$

Supongamos que existen dos soluciones  $x_1, x_2$  entonces tendremos:

$$\begin{aligned} y &= a^{x_1} \pmod{n} & 1 \leq x_1 < \Phi(n) \\ y &= a^{x_2} \pmod{n} & 1 \leq x_2 < \Phi(n) \end{aligned}$$

o lo que es lo mismo:

$$\begin{aligned} x_1 &= \log_a y \pmod{n} \\ x_2 &= \log_a y \pmod{n} \end{aligned}$$

Si restamos ambas soluciones módulo  $n$ :

$$x_1 - x_2 = \log_a y - \log_a y = 0$$

con lo que tenemos que  $x_1 - x_2 = \lambda \cdot n$  con  $\lambda \in \mathbb{Z}$ . Tenemos entonces que  $x_1 = x_2 + \lambda \cdot n$  con lo cual llegamos a contradicción ya que tanto  $x_1$  como  $x_2$  son menores que  $\Phi(n) < n$ . Luego no pueden existir  $x_1$  y  $x_2$  en las condiciones dadas verificando la ecuación, es decir  $y$  no puede tener dos logaritmos discretos.

Hemos demostrado que no pueden existir dos logaritmos discretos, pero no hemos demostrado la existencia de solución, es decir la existencia del logaritmo discreto. Veamos ahora la existencia del logaritmo discreto.

Por definición el logaritmo discreto es  $x = \log_a y$  y para que exista solución  $y$  debe ser positivo y no nulo. Que  $y$  sea positivo no plantea ningún problema, luego el único problema es que  $y$  sea no nulo. Pero como tenemos que  $y = a^x$  con  $0 \leq x < \Phi(n)$  y  $a \in (\mathbb{Z}/n)^*$ , el cual es un grupo multiplicativo al cual no pertenece el 0, tenemos que existe el logaritmo discreto. ■

Tenemos que la función exponencial definida por la ecuación (10.1) es una función “facil” de calcular, pero conocidos  $a, y, n$  encontrar  $x = \log_a y \pmod{n}$ , el logaritmo discreto, es difícil de calcular ya que no se conoce ningún algoritmo capaz de hacerlo en tiempo polinomial, mientras que la exponenciación se puede realizar en  $2 \cdot \lceil \log_2 n \rceil$  multiplicaciones módulo  $n$ . Luego la exponenciación se puede resolver en tiempo polinomial.

### 10.1.2. Algoritmo de Exponenciación Rápida

Cualquier sistema criptográfico que utilice este tipo de funciones necesitará calcular exponenciaciones modulares. Para calcular  $a^b$  el algoritmo usual es multiplicar  $a$  por sí mismo  $b$  veces, pero cuando  $b$  es un entero grande este algoritmo es poco efectivo. Podemos utilizar otro algoritmo más optimizado para calcular  $a^b$ .

Sea  $(b_0, b_1, \dots, b_n)_2$  la expresión en binario de  $b_{10}$ :

$$b_{10} = b_0 \cdot 2^0 + b_1 \cdot 2^1 + \dots + b_n \cdot 2^n \text{ con } b_i \in \{0, 1\}$$

utilizando esto tendremos que:

$$a^b = 2^{b_0 \cdot 2^0 + b_1 \cdot 2^1 + \dots + b_n \cdot 2^n} = \prod_{i=0}^n a^{2^i \cdot b_i}$$

Para calcular  $a^b$  sólo tendremos que multiplicar los  $a^{2^i}$  correspondientes a los  $b_i$  que sean 1. Además tenemos que  $a^{2^i} = (a^{2^{i-1}})^2$ , con lo que partiendo de  $a$  podemos calcular el valor siguiente de la sucesión  $\{a^{2^i}\}_i$  elevando al cuadrado el termino anterior.



# Capítulo 11

## El Algoritmo RSA

El algoritmo RSA es el algoritmo de clave pública más utilizado en la actualidad, también es el más sencillo tanto en cuanto a comprensión se refiere como a implementación. Su nombre proviene de sus tres inventores: Ron Rivest, Adi Shamir y Leonard Adleman. Desde su nacimiento nadie ha conseguido demostrar que el método es inseguro.

Este algoritmo se basa en la dificultad para factorizar grandes números. Tanto la clave pública como la privada se obtienen como el producto de dos números primos “grandes”, de unos 200 dígitos cada uno.

### 11.1. Elección de Claves en el Algoritmo RSA

El RSA es un algoritmo de clave pública con lo cual tendremos dos claves para cifrar y descifrar:

- La clave pública, la cual es conocida por todos los usuarios y es utilizada para codificar mensajes y mandárselos al usuario propietario de dicha clave. A esta clave la denotaremos como  $K_{Pb}$ .
- La clave privada, la cual no es pública y únicamente conoce el propietario de la clave. Esta clave se utiliza para descifrar los mensajes cifrados con la clave pública. A esta clave la denotaremos como  $K_{Pv}$ .

Para establecer las claves el usuario elija dos números  $p, q$  primos y relativamente grandes, unos 200 dígitos, entonces llamaremos  $n$  al siguiente número  $n = p \cdot q$ .

### 11.1.1. Elección de la Clave Privada, $K_{Pv}$

Buscaremos un número  $e$  tal que sea primo con  $\Phi(n) = (p-1) \cdot (q-1)$ . Dado que  $e$  y  $\Phi(n)$  son primos entre sí tendremos que existe  $d$  tal que:

$$e \cdot d \equiv 1 \pmod{\Phi(n) = (p-1) \cdot (q-1) = n + 1 - p - q}$$

$d$  se puede calcular mediante el algoritmo de Euclides.

Entonces la clave privada será  $K_{Pv} = (d, n)$ .

### 11.1.2. Elección de la Clave Pública, $K_{Pb}$

La clave pública será  $K_{Pb} = (e, n)$ .

Cualquier usuario que conozca nuestra clave pública sabe que:

$$e \cdot d \equiv 1 \pmod{\Phi(n)}$$

pero como desconoce la factorización en números primos de  $n$  el cálculo de  $d$  es prácticamente imposible realizarlo por fuerza bruta ya que  $n$  es un número de unos 400 dígitos.

### 11.1.3. Datos Privados

Los datos privados serán:

- La descomposición en números primos del número  $n$ , es decir los primos  $p$  y  $q$ .
- El valor  $\Phi(n)$ .
- El número  $d$  el cual verifica:

$$e \cdot d \equiv 1 \pmod{\Phi(n)}$$

Cualquiera que conozca estos datos podrá descifrar los mensajes del propietario de esta clave.

### 11.1.4. Datos Públicos

Los datos públicos serán:

- El número  $n$ .
- El número  $e$  el cual verifica:

$$e \cdot d \equiv 1 \pmod{\Phi(n)}$$

Cualquiera que conozca estos datos podrá mandar mensajes cifrados al usuario propietario de esta clave, siendo este el único capaz de descifrar el mensaje<sup>1</sup>.

## 11.2. Descripción del Algoritmo

Tenemos  $n = p \cdot q$  y las claves  $K_{P_b} = (n, e)$  y  $K_{P_v} = (n, d)$ . A continuación elegimos un alfabeto, por ejemplo el alfabeto de 36 letras de la página 129. Sea  $N$  la cantidad de símbolos que tiene el alfabeto utilizado. Cifraremos el mensaje por bloques, luego elegiremos la longitud de cada bloque así como la longitud en la que vamos a cifrar dicho bloque.

Matemáticamente la función de encriptación será:

$$\mathbb{Z}/N^k \longrightarrow \mathbb{Z}/N^l$$

y la desencriptación será:

$$\mathbb{Z}/N^l \longrightarrow \mathbb{Z}/N^k$$

donde  $k$  será la longitud del bloque que vamos a cifrar y  $l$  la longitud del bloque cifrado.  $k$  deberá verificar que  $N^k < n$ .

---

<sup>1</sup>Siempre cuando sea el único que conoce los datos privados.

### 11.2.1. Procedimiento de Encriptación

Sea  $T_0T_1 \dots T_{g-1}$  un texto que queremos cifrar. Lo primero que hacemos es asignar a cada  $T_i$  un número según un alfabeto:

$$a_0, a_1, \dots, a_{g-1}$$

seguidamente dividimos la expresión anterior en bloques de  $k$  números y procederemos con cada bloque de la siguiente manera:

$$(a_0, a_1, \dots, a_{k-1}) \in \mathbb{Z}/N^k \text{ con } a_i \in \mathbb{Z}/N$$

luego podemos considerar a  $(a_0, a_1, \dots, a_{k-1})$  como la expresión en base  $N$  de:

$$a_0 \cdot N^0 + a_1 \cdot N + \dots + a_{k-1} \cdot N^{k-1} = m < n$$

a continuación hacemos:

$$C = m^e \text{ mod } n$$

seguidamente transformamos  $C$  en un elemento de  $\mathbb{Z}/N^l$ :

$$C = c_0 \cdot N^0 + c_1 \cdot N + \dots + c_{l-1} \cdot N^{l-1} = (c_0, c_1, \dots, c_{l-1}) \in \mathbb{Z}/N^l$$

Hemos cifrado el bloque  $(a_0, a_1, \dots, a_{k-1})$  como  $(c_0, c_1, \dots, c_{l-1})$ .

### 11.2.2. Procedimiento de Desciframiento

Supongamos que recibimos  $(c_0, c_1, \dots, c_f)$ , entonces para descifrarlo primero lo dividiremos en bloques de longitud  $l$ ,  $(c_0, c_1, \dots, c_{l-1})$  y para cada bloque haremos:

$$C = c_0 \cdot N^0 + c_1 \cdot N + \dots + c_{l-1} \cdot N^{l-1} = (c_0, c_1, \dots, c_{l-1}) \in \mathbb{Z}/N^l$$

a continuación calculamos:

$$m = C^d \text{ mod } n$$

ahora calculamos la expresión de  $m$  en base  $N$ :

$$m = a_0 \cdot N^0 + a_1 \cdot N + \dots + a_{k-1} \cdot N^{k-1}$$

entonces el mensaje transmitido será:

$$T_0T_1 \dots T_{k-1}$$

donde  $T_i$  es la letra correspondiente a  $a_i$  según el alfabeto utilizado.

## 11.3. Ejemplo Práctico del Algoritmo RSA

Consideremos  $p = 281$ ,  $q = 167$ , con lo cual:

$$\begin{aligned}n &= 281 \cdot 167 = 46,921 \\ \Phi(46,921) &= (281 - 1) \cdot (167 - 1) = 46,480\end{aligned}$$

Busquemos  $e$  y  $d$  tal que  $e \cdot d \equiv 1 \pmod{\Phi(46,921)}$ :

$$\begin{aligned}e &= 39,423 \\ d &= 26,767\end{aligned}$$

Luego las claves serán:

$$\begin{aligned}K_{Pb} &= (46,921, 39,423) \\ K_{Pv} &= (46,921, 26,767)\end{aligned}$$

Supongamos que queremos mandar el mensaje “HOLA” utilizando el alfabeto de 36 símbolos, en la página [129](#), cifrando bloques de dos letras en bloques de tres.

### 11.3.1. Procedimiento de Encriptación

Según el alfabeto tendremos que:

$$\text{HOLA} = (17, 24, 21, 10)$$

Como vamos a cifrar utilizando bloques de dos letras, los bloques a cifrar serán:

$$(17, 24) \quad \text{y} \quad (21, 10)$$

Estos bloques entendidos como elementos de  $(\mathbb{Z}/36)^2$  son:

$$\begin{aligned}(17, 24) &= 17 \cdot 36^0 + 24 \cdot 36 = 881 \\ (21, 10) &= 21 \cdot 36^0 + 10 \cdot 36 = 381\end{aligned}$$

tendremos entonces:

$$\begin{aligned}881^{39,423} &\equiv 45,840 \pmod{46,927} \\ 381^{39,423} &\equiv 26,074 \pmod{46,927}\end{aligned}$$

Ahora expresamos estos números como elementos de  $(\mathbb{Z}/36)^3$ :

$$\begin{aligned}45,840 &= 12 \cdot 36^0 + 13 \cdot 36 + 35 \cdot 36^2 = (12, 13, 35) \in (\mathbb{Z}/36)^3 \\ 26,074 &= 10 \cdot 36^0 + 4 \cdot 36 + 20 \cdot 36^2 = (10, 4, 20) \in (\mathbb{Z}/36)^3\end{aligned}$$

Según el alfabeto utilizado tendremos:

$$\begin{aligned}(12, 13, 35) &\implies \text{CDZ} \\ (10, 4, 20) &\implies \text{A4K}\end{aligned}$$

Luego el mensaje “HOLA” cifrado es “CDZA4K”.

### 11.3.2. Procedimiento de Descriptación

Supongamos que hemos recibido el mensaje encriptado “CDZA4K”, vamos a descriptarlo. Como sabemos que el mensaje ha sido encriptado encriptando bloques de dos letras en bloques de tres tendremos que dividirlo en bloques de tres letras y asignar a cada letra un número de acuerdo con el alfabeto utilizado:

$$\begin{aligned}\text{CDZ} &= (12, 13, 35) \\ \text{A4K} &= (10, 4, 20)\end{aligned}$$

Considerando estos vectores como elementos de  $(\mathbb{Z}/36)^3$  tendremos:

$$\begin{aligned}(12, 13, 35) &= 12 \cdot 36^0 + 13 \cdot 36 + 35 \cdot 36^2 = 45,840 \\ (10, 4, 20) &= 10 \cdot 36^0 + 4 \cdot 36 + 20 \cdot 36^2 = 26,074\end{aligned}$$

tendremos entonces que:

$$\begin{aligned}45,840^{26,767} &\equiv 881 \pmod{46,927} \\ 26,074^{26,767} &\equiv 381 \pmod{46,927}\end{aligned}$$

expresamos estos números como elementos de  $(\mathbb{Z}/36)^2$ :

$$\begin{aligned}881 &= 17 \cdot 36^0 + 24 \cdot 36 = (17, 24) \in (\mathbb{Z}/36)^2 \\ 381 &= 21 \cdot 36^0 + 10 \cdot 36 = (21, 10) \in (\mathbb{Z}/36)^2\end{aligned}$$

Consultando el alfabeto utilizado tendremos que:

$$\begin{aligned}(17, 24) &\implies \text{HO} \\ (21, 10) &\implies \text{LA}\end{aligned}$$

Luego el texto encriptado “CDZA4K” es “HOLA”.

## 11.4. Justificación Matemática del Algoritmo RSA

Sea  $n = p \cdot q$ , con  $p$  y  $q$  primos y sean  $e$  y  $d$  dos números verificando:

$$e \cdot d \equiv 1 \pmod{\Phi(n)} \quad (11.1)$$

la encriptación la hemos llevado a cabo haciendo:

$$c = m^e \pmod{n}$$

mientras que la desencriptación la hemos llevado a cabo haciendo:

$$m = c^d \pmod{n}$$

pero esta última expresión no la hemos justificado:

$$c^d = (m^e)^d \stackrel{(1)}{=} m^{k \cdot \Phi(n) + 1} = m^{k \cdot \Phi(n)} \cdot m \stackrel{(2)}{=} m$$

(1) por (11.1).

(2) por la proposición 8.3, en la página 70.

## 11.5. Seguridad del Algoritmo RSA

El algoritmo RSA basa su seguridad en la complejidad del logaritmo discreto, ya que no existe ningún algoritmo conocido que pueda realizarlo en tiempo polinomial. Además nadie ha demostrado que se pueda elaborar un algoritmo para descifrar mensajes cifrados con RSA sin conocer la clave privada.

La seguridad de este algoritmo se basa en mantener en secreto  $p$  y  $q$ , así como  $\Phi(n)$ , ya que conocer  $\Phi(n)$  equivale a conocer  $p$  y  $q$ .

$$\begin{aligned} n &= p \cdot q \\ \Phi(n) &= p + q - (n + 1) \Rightarrow p + q = n + 1 - \Phi(n) \end{aligned}$$

$p$  y  $q$  son las raíces de la ecuación:

$$x^2 + (n + 1 - \Phi(n)) \cdot x + n = 0$$

luego si conocemos  $\Phi(n)$  podemos calcular  $p$  y  $q$  resolviendo la ecuación de segundo grado anterior<sup>2</sup>, lo cual nos dará  $p$  y  $q$  y habremos roto la seguridad del sistema.

---

<sup>2</sup>Recordar que  $n$  es un dato público.

### 11.5.1. Algoritmo RSA y el Algoritmo de Shor

Actualmente se está investigando sobre ordenadores y computación cuántica. Existe un algoritmo, el algoritmo de Shor, un algoritmo cuántico, el cual ejecutado sobre un ordenador cuántico es capaz de factorizar cualquier número en tiempo polinomial. Debido a esto en el momento en el que se logren desarrollar este tipo de ordenadores cualquier método criptográfico basado en la complejidad del logaritmo discreto será un método inseguro, ya que existieran herramientas que nos permitirán romper dichos métodos en tiempo polinomial.

Se puede encontrar más información sobre este tema en:

P.W.Shor, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, editado por S.Goldwasser(IEEE Computer Society Press, CA), en 1,994.

### 11.5.2. Claves Débiles en RSA

Existen ciertos casos para los cuales el algoritmo RSA deja el mensaje original sin cambio alguno:

$$m^e = m \pmod n$$

Se puede comprobar que, siendo  $n = p \cdot q$  y  $e$  el exponente para codificar

$$\sigma_n = (1 + \text{M.C.D.}(e - 1, p - 1))(1 + \text{M.C.D.}(e-1, q-1))$$

es el número de valores de  $m$  que quedan igual al ser codificados. Si hacemos que  $p = 1 + 2 \cdot p'$  y  $q = 1 + 2 \cdot q'$ , con  $p'$  y  $q'$  primos, entonces  $\text{M.C.D.}(e - 1, p - 1)$  puede valer 1, 2 o  $p'$ , análogamente ocurre con  $q'$ . Los valores posibles de  $\sigma_n$  serán entonces 4, 6, 9,  $2 \cdot (p' + 1)$ ,  $2 \cdot (q' + 1)$ ,  $3 \cdot (p' + 1)$ ,  $3 \cdot (q' + 1)$  y  $(p' + 1) \cdot (q' + 1)$ . Los cinco últimos son bastante improbables, luego no deben preocuparnos. Como medida de precaución se puede calcular  $\sigma_n$  a la hora de generar las llaves pública y privada.

### 11.5.3. Claves Demasiado Cortas

Para uso del RSA se recomiendan claves de, al menos, 1,024 bits. Hasta hace poco se utilizaban claves de 512 bits, pero en Mayo de 1,999, Adi Shamir presentó el dispositivo *Twinkle*, un ingenio capaz de factorizar números de manera muy rápida, aprovechando los últimos avances en la optimización de algoritmos específicos para esta tarea. Este dispositivo, aún no construido, podría ser incorporado en ordenadores de bajo coste y pondría en serio peligro los mensajes cifrados con claves de menos de 512 bits.

Teniendo en cuenta los avances de la tecnología, y suponiendo que el RSA no sea roto analíticamente, deberemos escoger la longitud de la clave en función del tiempo que queramos que nuestra información permanezca en secreto.

### 11.5.4. Ataques de Intermediario

Este tipo de ataque puede darse a cualquier algoritmo de clave pública. Supongamos que el usuario A quiere comunicarse con el usuario B. El usuario A solicita la clave pública del usuario B,  $K_B$ , entonces un tercer usuario, el usuario C, intercepta su comunicación y suministra a A su clave pública  $K_C$ . Entonces el usuario A tiene la clave pública  $K_C$  creyendo que es la de el usuario B. Manda un mensaje al usuario B codificado con  $K_C$ , este mensaje es interceptado por C, descodificado, vuelto a codificar a conveniencia de C con la clave  $K_B$ , clave pública de B, y enviado a B suplantando la identidad de A. Ni A ni B se han dado cuenta de que sus mensajes han sido interceptados y modificados.

La única forma de evitar este tipo de ataques consiste en asegurar a A que la clave pública de B es auténtica. Para esto existen organismos, de confianza, que certifican las claves públicas de los usuarios, con lo cual no se puede llevar a cabo este tipo de ataque si los usuarios utilizan algún tipo de certificación de este tipo.

### 11.5.5. Ataques de Texto Plano Escogido

Existe una familia de ataques a RSA que explotan la posibilidad de que un usuario codifique y firme un único mensaje empleando el mismo par de llaves. Para que el ataque surta efecto, la firma debe hacerse codificando el mensaje completo, no el resultado de una función resumen o función hash sobre él. Por ello se recomienda que las firmas digitales se lleven a cabo siempre sobre una función resumen del mensaje, nunca sobre el mensaje.

Otro tipo de ataque con texto plano escogido podría ser el siguiente: para falsificar una firma sobre un mensaje  $m$ , se pueden calcular dos mensajes individuales  $m_1$  y  $m_2$ , aparentemente inofensivos, tales que  $m_1 \cdot m_2 = m$  y enviarlos a la víctima para que los firme. Entonces obtendríamos  $m_1^d$  y  $m_2^d$ . Aunque desconozcamos  $d$  tendremos:

$$m_1^d \cdot m_2^d = m^d \pmod{n}$$

obtendremos el mensaje  $m$  firmado.

### 11.5.6. Ataques de Módulo Común

Podría pensarse que, una vez generados  $p$  y  $q$ , será más rápido generar tantos pares de llaves como queramos, en lugar de tener que emplear dos números primos diferentes en cada caso. Sin embargo, si lo hacemos así, un atacante podrá descifrar nuestros mensajes sin necesidad de llave privada. sea  $m$  el texto plano, que codificamos empleando dos claves de cifrado diferentes  $e_1$  y  $e_2$ . Los criptogramas que obtenemos son:

$$\begin{aligned} c_1 &= m^{e_1} \pmod{n} \\ c_2 &= m^{e_2} \pmod{n} \end{aligned}$$

el atacante conocería  $n$ ,  $e_1$ ,  $e_2$ ,  $c_1$  y  $c_2$ . Si  $e_1$  y  $e_2$  son primos entre sí, algo muy probable, la ecuación diofántica:

$$r \cdot e_1 + s \cdot e_2 = 1$$

tiene solución, con lo cual:

$$c_1^r \cdot c_2^s = m^{e_1 \cdot r} \cdot m^{e_2 \cdot s} = m^{r \cdot e_1 + s \cdot e_2} = m^1 \pmod{n}$$

Como consecuencia de esto se deben generar  $p$  y  $q$  diferentes para cada par de claves.

### **11.5.7. Firmas digitales y RSA**

Con el algoritmo RSA nunca se debe firmar un mensaje después de cifrarlo, debe firmarse primero. Existen ataques que aprovechan mensajes primero cifrados y luego firmados, aunque se empleen funciones resumen.



Parte V

Criptografía con Curvas  
Elípticas



# Capítulo 12

## Curvas Elípticas

En lo sucesivo cuando hablemos de un cuerpo  $\mathbb{K}$  supondremos uno de los siguientes casos:

- $\mathbb{K} = \mathbb{C}$ .
- $\mathbb{K} = \mathbb{R}$ .
- $\mathbb{K} = \mathbb{Q}$ .
- $\mathbb{K} = \mathbb{F}_q$  con  $q = p^r$ ,  $p$  primo<sup>1</sup> y  $r \in \mathbb{N}$ .

### 12.1. Definición de Curvas Elípticas

Una curva elíptica es una curva plana, en el plano, definida por la siguiente ecuación:

$$y^2 = x^3 + a \cdot x + b$$

donde  $a, b \in \mathbb{K}$ .

#### **Definición 12.1 (Curva Elíptica)**

*Llamaremos Curva Elíptica al siguiente conjunto:*

$$E = \{ (x, y) \in \mathbb{K} \times \mathbb{K} \text{ tales que } y^2 = x^3 + a \cdot x + b \} \cup \{ \mathcal{O} \}$$

*El punto  $\mathcal{O}$  recibe el nombre de punto del infinito de la curva elíptica, este punto no está en el plano. Este punto se introduce para poder dotar a la curva de estructura de grupo.*

---

<sup>1</sup>Distinto de 2 y 3.

**Definición 12.2 (Discriminante de una Curva Elíptica)**

Llamaremos *discriminante de una Curva Elíptica* y lo denotaremos por  $\Delta$ :

$$\Delta = 4 \cdot a^3 + 27 \cdot b^2 \in \mathbb{K}$$

Supondremos que  $x^3 + a \cdot x + b$  no tiene raíces múltiples.  $x^3 + a \cdot x + b$  no tiene raíces múltiples  $\iff \Delta = 4 \cdot a^3 + 27 \cdot b^2 \neq 0$ .

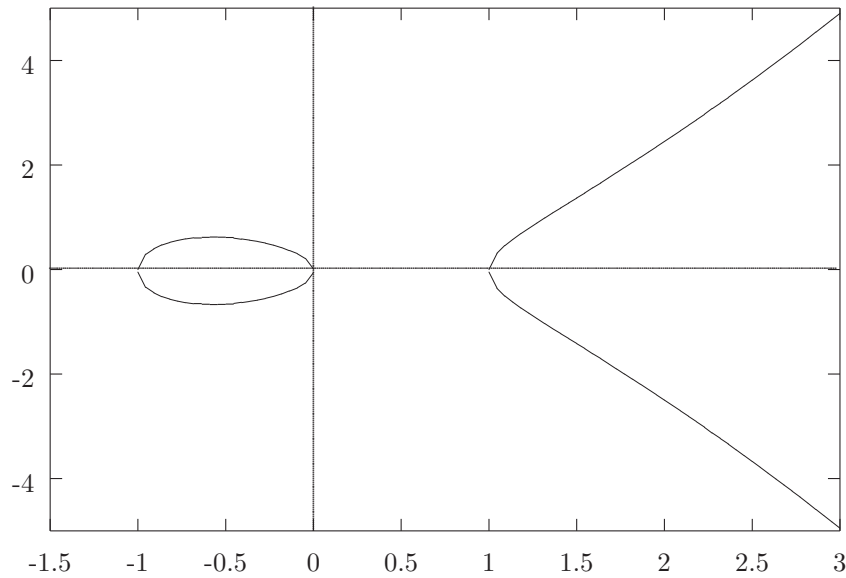


Figura 12.1: Ejemplo de curva elíptica sobre  $\mathbb{R}$ .

## 12.2. Estructura de grupo de las Curvas Elípticas

Sea  $E$  la Curva Elíptica definida por:

$$y^2 = x^3 + a \cdot x + b$$

vamos a definir la siguiente ley de grupo en  $E$ :

$$\begin{array}{ccc} E \times E & \xrightarrow{+} & E \\ (P, Q) & \longrightarrow & P + Q \end{array}$$

1.  $P + \mathcal{O} = P \forall P \in E$ .  $\mathcal{O}$  es el elemento neutro.
2. Si  $P \neq \mathcal{O}$  y  $(x, y)$  son las coordenadas de  $P$ , se define  $-P$  como el punto de  $E$  de coordenadas  $(x, -y)$ . Definimos  $-\mathcal{O}$  como  $\mathcal{O}$ .
3. Si  $P, Q \in E$  tales que  $P, Q \neq \mathcal{O}$  se define  $P + Q$  del siguiente modo:
  - a)  $P \neq Q$ , donde  $P$  y  $Q$  tienen distinta coordenada  $x$ .

Sea  $l = \langle P, Q \rangle$  la única recta del plano que pasa por  $P$  y  $Q$ . Tendremos que  $l \cap E$  son tres puntos:

$$l \cap E = \{ P, Q, R \}$$

Sean  $(x, y) = R$  las coordenadas de  $R$ , entonces se define la suma de  $P$  y  $Q$  como:

$$P + Q = -R$$

donde  $-R$  es el punto cuyas coordenadas son  $(x, -y)$ . La figura 12.2 muestra la interpretación geométrica de esta suma.

- b)  $P \neq Q$ , donde  $P$  y  $Q$  tienen la misma coordenada  $x$ .

En este caso tendremos que  $Q = -P$ , con lo cual tendremos:

$$P + Q = P + (-P) = \mathcal{O}$$

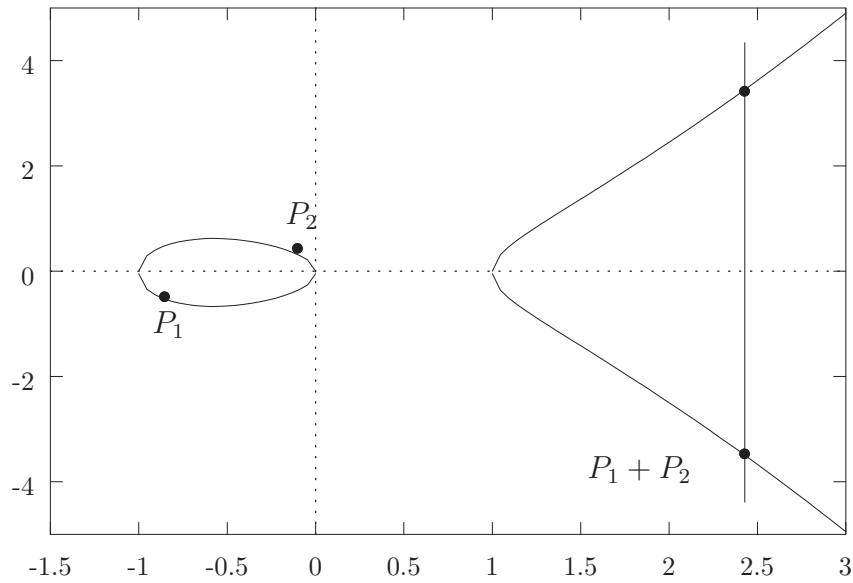


Figura 12.2: Ejemplo de la suma de dos elementos.

c)  $P = Q$

Cuando  $P = Q$  hacemos tender el punto  $Q$  a  $P$  entonces la recta  $l$  es la recta tangente en el punto  $P$ , luego:

$$l \cap E = \{ P, R \}$$

con lo que definimos la suma de  $P$  consigo mismo como:

$$P + P = -R$$

La figura 12.3 muestra la interpretación geométrica de esta suma.

### Definición 12.3 (Producto por un escalar)

Dado  $P \in E$  y  $n \in \mathbb{N}$  podemos definir  $n \cdot P$ :

$$n \cdot P = P + \dots + P$$

### Definición 12.4 ( $E[n]$ )

$$E[n] = \{ P \in E \text{ tales que } n \cdot P = \mathcal{O} \}$$

Tenemos que  $E[n] \subset E$  y además es un subgrupo finito.

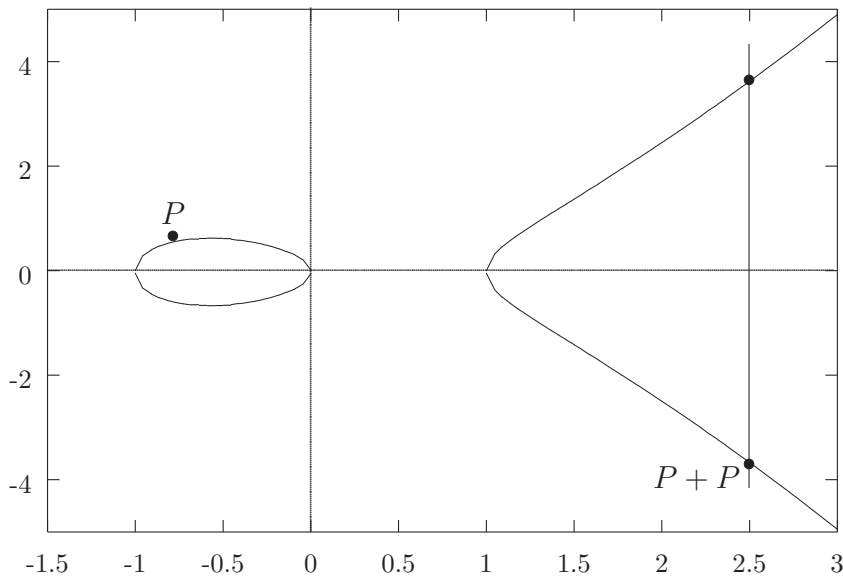


Figura 12.3: Ejemplo de la suma de un elemento consigo mismo.

Si  $\mathbb{K} = \mathbb{C}$  tenemos que:

$$E[n] = \mathbb{Z}/n \times \mathbb{Z}/n$$

lo cual tiene relación con el hecho de que:

$$\mu_n \simeq \mathbb{Z}/n$$

donde  $\mu_n$  son las raíces  $n$ -esimas de la unidad y se tiene que  $E[n]=n^2$ .

**Teorema 12.1 (Estructura de grupo de las Curvas Elípticas)**  
 *$(E, +)$  es un grupo abeliano. Donde  $E$  esta definida por:*

$$y^2 = x^3 + a \cdot x + b$$

**Demostración:**

1. En primer lugar veamos que si  $P = (x, y) \in E$  entonces el punto  $-P = (x, -y) \in E$ :

Como  $P \in E$  entonces tendremos:

$$y^2 = x^3 + a \cdot x + b \Rightarrow y^2 - x^3 - a \cdot x - b = 0$$

entonces tendremos que:

$$(-y)^2 - x^3 - a \cdot x - b = 0 \Rightarrow (-y)^2 = x^3 + a \cdot x + b$$

luego  $-P = (x, -y) \in E$ .

2. En segundo lugar veamos que  $P + Q \in E \forall P, Q \in E$ :

▪  $P$  o  $Q$  son  $\mathcal{O}$ :

$$\mathcal{O} + Q = Q \in E$$

$$P + \mathcal{O} = P \in E$$

$$\mathcal{O} + \mathcal{O} = \mathcal{O} \in E$$

▪  $P, Q \neq \mathcal{O}$ :

Se deduce inmediatamente de las definiciones y del hecho de que si  $P \in E$  entonces  $-P \in E$ .

3.  $\mathcal{O}$  es el elemento neutro y la suma de un punto  $P$  con su opuesto  $-P$  es el elemento neutro.

Si sumamos un elemento con su opuesto se puede ver que la recta que los une no corta a la curva elíptica más que en dos puntos, no en tres. Es por esta razón que se introduce  $\mathcal{O}$  como punto del infinito para dotar al conjunto  $E$  de estructura de grupo. Luego por definición la suma de un punto con su opuesto es  $\mathcal{O}$ .

Un ejemplo de esto se puede ver en la figura [12.4](#).

4. El elemento opuesto es único:

Sea  $P = (x, y) \in E$  un punto de la curva elíptica, entonces por definición su elemento opuesto será  $-P = (x, -y)$ , al estar la coordenada  $y$  determinada de forma única entonces la segunda coordenada de  $-P$  está determinada también de forma única.

■

### Corolario 12.1

$|E| \cdot P = \mathcal{O} \forall P \in E$ , donde  $|E|$  es el número de puntos de  $E$ .

## 12.3. Curvas Elípticas sobre Cuerpos Finitos

Los cuerpos finitos sobre los que vamos a considerar las curvas elípticas serán de la forma  $\mathbb{F}_q$ , con  $q = p^r$ , donde  $p \neq 2, 3$  y primo.

Puesto que en los cuerpos finitos tenemos un número finito de puntos entonces cualquier curva sobre un cuerpo finito tiene un número finito de puntos distintos.

### Teorema 12.2 (de Hasse)

Sea  $E$  una curva elíptica sobre  $\mathbb{F}_q$  con  $q = p^r$ . Si denotamos por  $N(E)$  al número de puntos de dicha curva sobre  $\mathbb{F}_q$  entonces tenemos que:

$$|N(E) - q - 1| \leq 2 \cdot \sqrt{q}$$

Este resultado es analogo para curvas elípticas que verifiquen la hipótesis de Riemann. Este resultado nos permite acotar el número de puntos que tiene una curva elíptica sobre un cuerpo finito.

Existe un algoritmo para el calculo de  $N(E)$ , el tiempo de computo de dicho algoritmo es  $O(\log p^r)$ , es un algoritmo polinomico y su autor es René Schoof.

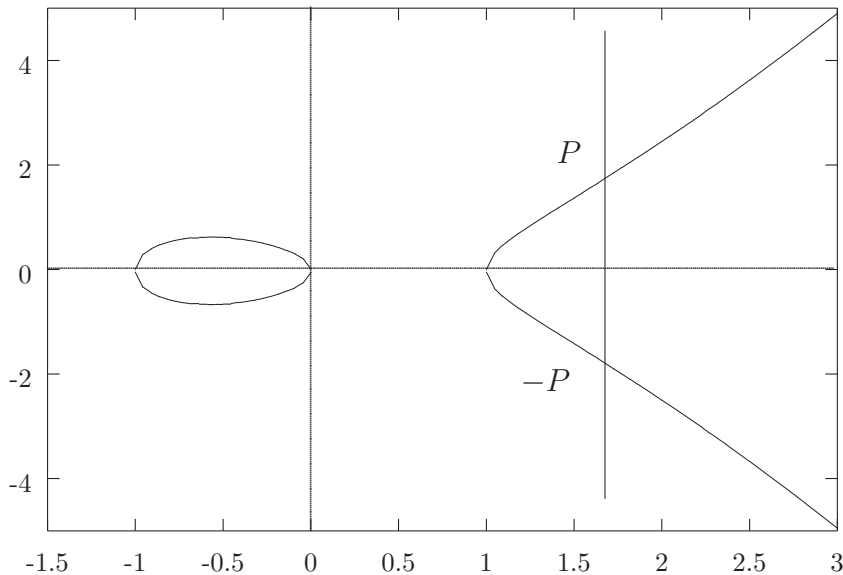


Figura 12.4: Ejemplo de la suma de un elemento y su opuesto.

### 12.3.1. Orden de las Curvas Elípticas definidas sobre Cuerpos de Característica 2

#### Teorema 12.3 (de Weil)

Sea  $E$  una curva elíptica definida sobre  $\mathbb{F}_{p^m}$  con  $p$  primo y  $m \in \mathbb{Z}$ , entonces se verifica:

$$N(E) = p^m + 1 - (\alpha^m + \beta^m)$$

donde  $\alpha$  y  $\beta$  son las raíces complejas de la ecuación:

$$x^2 + (N(E)' - p - 1) \cdot x + p = 0$$

y  $N(E)'$  es el orden de la curva  $E$  definida sobre  $\mathbb{F}_{m'}$  con  $p$  primo y  $m' \in \mathbb{Z}$  tal que  $m' \lll m$ .

Consideremos la siguiente curva elíptica:

$$y^2 + x \cdot y = x^3 + x^2 + 1$$

vamos a calcular su orden en el cuerpo  $\mathbb{F}_{2^8}$ .

1. Consideremos dicha curva en otro cuerpo de característica 2 “bastante menor” que  $\mathbb{F}_{2^8}$ , por ejemplo  $\mathbb{F}_{2^1}$ . En  $\mathbb{F}_{2^1}$  dicha curva únicamente tiene dos puntos:

- $(0, 1)$ .
- $\mathcal{O}$ .

Luego  $N(E)' = 2$ .

2. Calculamos  $\alpha$  y  $\beta$ , que serán solución de la ecuación:

$$x^2 + (N(E)' - p - 1) \cdot x + p = x^2 - x + 2 = 0$$

$$\alpha = \frac{1 + \sqrt{7} \cdot i}{2}$$

$$\beta = \frac{1 - \sqrt{7} \cdot i}{2}$$

3. Calculamos  $N(E)$ :

$$N(E) = 2^8 + 1 - \left( \frac{1 - \sqrt{7} \cdot i}{2} \right)^8 - \left( \frac{1 + \sqrt{7} \cdot i}{2} \right)^8 = 288$$

Luego la curva  $y^2 + x \cdot y = x^3 + x^2 + 1$  tiene 288 puntos sobre  $\mathbb{F}_{2^8}$ .

### 12.3.2. Cálculos en Coordenadas Projectivas

Uno de los mayores inconvenientes que tiene la suma de puntos en curvas elípticas es el calculo de inversos modulares, el cual requiere más tiempo de computo que las sumas y multiplicaciones modulares. *Chudnovsky* y *Chudnovsky* propusieron un método para sumar puntos de curvas elípticas, sin necesidad de realizar inversiones modulares, mediante coordenadas proyectivas.

Lo primero que tenemos que hacer es transformar la ecuación que define la curva elíptica mediante el cambio de las coordenadas afines  $(x, y)$  a las coordenadas proyectivas  $(x', y', z')$  definido por:

$$(x, y) \mapsto (x', y') = (x'/z'^2, y'/z'^3)$$

con  $z' \in \mathbb{F}_q$ . La ecuación de la curva en coordenadas proyectivas será:

$$y'^2 = x'^3 + a \cdot z'^4 \cdot x' + b \cdot z'^6$$

Tenemos que cada punto  $(x, y)$  de la curva original, en coordenadas afines, tiene múltiples representaciones en coordenadas proyectivas  $(x', y', z')$  dependiendo del valor que tome  $z'$ . Una representación trivial del punto  $(x, y)$ , en coordenadas afines, a coordenadas proyectivas es  $(x, y, 1)$ . Cada punto en coordenadas proyectivas sólo tiene una única representación en coordenadas afines.

Sean  $P_1 = (x_1, y_1)$ ,  $P_2 = (x_2, y_2)$  dos puntos de la curva en coordenadas afines y sea  $P_3 = (x_3, y_3)$  su suma en coordenadas afines también. Sean  $P'_1 = (x'_1, y'_1, z'_1)$  y  $P'_2 = (x'_2, y'_2, z'_2)$  dos de las posibles representaciones en coordenadas proyectivas de los puntos  $P_1 = (x_1, y_1)$  y  $P_2 = (x_2, y_2)$ . Para calcular las coordenadas proyectivas  $P'_3 = (x'_3, y'_3, z'_3)$  del punto  $P_3 = (x_3, y_3)$  definimos los siguientes parámetros:

$$U_1 = x'_1 \cdot z'^2_2 \quad U_2 = x'_2 \cdot z'^2_1 \quad S_1 = y'_1 \cdot z'^3_2 \quad S_2 = y'_2 \cdot z'^3_1$$

y realizamos las siguientes operaciones:

- Si  $U_1 \neq U_2$  o  $S_1 \neq S_2$  se calcula  $W = z'_1 \cdot z'_2$  y se aplican las ecuaciones:

$$\begin{aligned} x'_3 &= (U_1 + U_2) \cdot (x'_1 \cdot x'_2 + a \cdot W^2) + 2 \cdot b \cdot W^4 - 2 \cdot y'_1 \cdot y'_2 \cdot W \\ y'_3 &= z'_2 \cdot (U_1 + 3 \cdot U_2) \cdot (a \cdot S_1 \cdot z'_1 - x'^2_1 \cdot y'_2) + \\ &\quad + z'_1 \cdot (3 \cdot U_1 + U_2) \cdot (x'^2_2 \cdot y'_1 - a \cdot S_2 \cdot z'_2) + 4 \cdot b \cdot W^3 \cdot (S_1 - S_2) \\ z'_3 &= U_2 - U_1 \end{aligned}$$

- Si  $U_1 = U_2$  y  $S_1 = S_2$  definimos los siguientes parámetros:

$$S = 4 \cdot x'_1 \cdot y_1'^2 \quad M = 3 \cdot x_1'^2 + a \cdot z_1'^4 \quad T = M^2 - 2 \cdot S$$

y aplicamos las siguientes ecuaciones:

$$\begin{aligned} x'_3 &= T \\ y'_3 &= M \cdot (S - T) - 8 \cdot y_1'^4 \\ z'_3 &= 2 \cdot y_1' \cdot z_1' \end{aligned}$$

Ahora ya tenemos las coordenadas proyectivas de  $P'_3$ :

$$P'_3 = (x'_3, y'_3, z'_3)$$

para obtener las coordenadas afines de  $P_3$ :

$$P_3 = (x_3, y_3) = \left( \frac{x'_3}{z'_3}, \frac{y'_3}{z'_3} \right)$$

Se puede ver que no es necesario el cálculo de ningún inverso modular, pero se realizan más multiplicaciones con respecto a las utilizadas en el cálculo con coordenadas afines. A pesar de esto este método es un 30 % más rápido que si usáramos coordenadas afines.

## 12.4. Como obtener los múltiplos de puntos de una Curva Elíptica

Muchas veces en criptografía es necesario calcular  $n \cdot P$  con  $n \in \mathbb{F}_q$  y  $P \in E$ , veamos algunos de los procedimientos utilizados para ello.

Supongamos que tenemos una curva elíptica definida por la ecuación:

$$y^2 = x^3 + a \cdot x + b$$

sobre  $\mathbb{F}_q$ , con  $q = p^r$ ,  $p$  primo y  $r \in \mathbb{N}$ .

### 12.4.1. Procedimiento de Izquierda a Derecha

Para calcular  $n \cdot P = Q$ , lo primero que tenemos que hacer es expresar  $n$  en binario:

$$n_{10} = (n_{k-1}, \dots, n_0)_2 \text{ con } n_i \in \{0, 1\}$$

El algoritmo es el siguiente:

1.  $Q = \mathcal{O}$ .
2. Para  $i = k - 1$  hasta  $i = 0$ :
  - a)  $Q = 2 \cdot Q$ .
  - b) Si  $b_i = 1$  entonces:
    - 1)  $Q = Q + P$ .
3.  $i = i - 1$  y vuelta al paso 2.
4. Obtención de  $Q$ .

#### Ejemplo 12.1 (Procedimiento de Izquierda a Derecha)

Sea la curva elíptica definida por la ecuación:

$$y^2 = x^3 - x - 563$$

en  $\mathbb{F}_{751}$ , y sea  $P = (484, 590)$  un punto de dicha curva. Supongamos que queremos calcular el punto  $Q = 20 \cdot (484, 590)$ .

El desarrollo en binario de 20 es el siguiente:

$$20_{10} = (1, 0, 1, 0, 0)_2$$

aplicando el algoritmo tendremos:

$i$	$h_i$	$Q$
5	-	$\mathcal{O}$
4	1	$2 \cdot \mathcal{O} + (484, 590) = (484, 590)$
3	0	$2 \cdot (484, 590) = (159, 124)$
2	1	$2 \cdot (159, 124) + (484, 590) = (172, 255) + (484, 590) = (514, 201)$
1	0	$2 \cdot (514, 201) = (79, 418)$
0	0	$2 \cdot (79, 418) = (317, 181)$

Luego  $20 \cdot (484, 590) = (317, 181)$ .

*El inconveniente de este método es que en cada suma elemental de puntos es necesario el cálculo de un inverso modular<sup>2</sup> para la pendiente de la recta definida por los sumandos. Una forma de evitar este inconveniente es utilizando las denominadas “coordenadas proyectivas”<sup>3</sup>.*

*Para la realización práctica de este ejemplo hemos utilizado el programa Mathematica usando el algoritmo de cálculo en coordenadas proyectivas para calcular tanto la suma de dos puntos distintos de la curva elíptica como para el cálculo de la suma de un punto consigo mismo.*

### 12.4.2. Funciones de Schoof

Siempre que  $n \cdot P \neq \mathcal{O}$  se tiene que:

$$n \cdot P(x, y) = \left( x - \frac{\Psi_{n-1} \cdot \Psi_{n+1}}{\Psi_n^2}, \frac{\Psi_{n+2} \cdot \Psi_{n-1}^2 - \Psi_{n-2} \cdot \Psi_{n+1}^2}{4 \cdot y \cdot \Psi_n^3} \right)$$

Las funciones  $\Psi_i$  se definen de forma recursiva:

$$\begin{aligned} \Psi_{-1}(x, y) &= -1 \\ \Psi_0(x, y) &= 0 \\ \Psi_1(x, y) &= 1 \\ \Psi_2(x, y) &= 2 \cdot y \\ \Psi_3(x, y) &= 3 \cdot x^4 + 6 \cdot a \cdot x^2 + 12 \cdot b \cdot x - a^2 \\ \Psi_4(x, y) &= 4 \cdot y \cdot (x^6 + 5 \cdot a \cdot x^4 + 20 \cdot b \cdot x^3 - 5 \cdot a^2 \cdot x^2 - 4 \cdot a \cdot b \cdot x - 8 \cdot b^2 - a^3) \end{aligned}$$

.....

$$\Psi_{2n}(x, y) = \frac{\Psi_n(\Psi_{n+2} \cdot \Psi_{n-1}^2 - \Psi_{n-2} \cdot \Psi_{n+1}^2)}{2 \cdot y} \text{ para } n = 3, 4, \dots$$

$$\Psi_{2n+1}(x, y) = \Psi_{n-2} \cdot \Psi_n^3 - \Psi_{n-1} \cdot \Psi_{n+1}^3 \text{ para } n = 3, 4, \dots$$

Este método no es recomendable ya que, en general, resulta algo más lento que el de izquierda a derecha descrito en el apartado anterior.

<sup>2</sup>El cálculo de inversos modulares tiene un mayor coste computacional que las sumas o multiplicaciones modulares.

<sup>3</sup>Apartado 12.3.2 en la página 111.

## 12.5. Sistema Criptográfico de Massey-Omura

Supongamos que tenemos dos usuarios  $A$  y  $B$  que desean comunicarse entre sí utilizando un sistema criptográfico basado en curvas elípticas.

Supongamos que tenemos:

$$y^2 = x^3 + a \cdot x + b$$

$E, \mathbb{F}_q$  con  $q = p^r$ , donde  $p \gg 0$  y  $p$  primo.

Los datos públicos serán los siguientes:

- $a, b \in \mathbb{F}_q$ .
- $N(E)$ , número de elementos de la curva elíptica.

Cada usuario elige un número  $e$  tal que:

$$\text{M.C.D } (e, N(E)) = 1 \iff e \in (\mathbb{Z}/N(E))^*$$

por el algoritmo de Euclides calculamos  $d = e^{-1} \pmod{N(E)}$ .

Los mensajes que queremos transmitir se identificarán con puntos  $P_m \in E$ .

$$E = \{ (x, y) \in \mathbb{F}_q \times \mathbb{F}_q \mid y^2 = x^3 + a \cdot x + b \}$$

La situación es la siguiente:

- La llave del usuario  $A$  es  $(e_A, d_A)$ .
- La llave del usuario  $B$  es  $(e_B, d_B)$ .

Sea  $m$  el mensaje que quiere mandar el usuario  $A$  al usuario  $B$ :

1. El usuario  $A$  identifica el punto  $P_m \in E$  que se corresponde con  $m$ .
2. El usuario  $A$  manda al usuario  $B$  el mensaje encriptado  $e_A \cdot P_m$ .
3. El usuario  $B$  manda al usuario  $A$  el mensaje  $e_B \cdot e_A \cdot P_m$ .
4. El usuario  $A$  manda al usuario  $B$  el mensaje  $d_A \cdot e_B \cdot e_A \cdot P_m$ .

Luego el usuario  $B$  recibe el mensaje  $m$  encriptado como  $d_A \cdot e_B \cdot e_A \cdot P_m$ , veamos como se obtiene el original:

$$\begin{aligned} e_A \cdot d_A &\equiv 1 \pmod{N(E)} \\ e_A \cdot d_A &= 1 + \lambda \cdot N(E) \text{ con } \lambda \in \mathbb{Z} \end{aligned}$$

aplicando esto tendremos:

$$\begin{aligned} d_A \cdot e_B \cdot e_A \cdot P_m &= e_A \cdot d_A \cdot e_B \cdot P_m = (1 + \lambda \cdot N(E)) e_B \cdot P_m = \\ &= e_B \cdot P_m + \lambda \cdot e_B \cdot N(E) \cdot P_m \stackrel{(*)}{=} e_B \cdot P_m \end{aligned}$$

(\*) ya que  $N(E) \cdot P_m = 0$  al ser  $N(E)$  el número de puntos de  $E$  y  $P_m \in E$ .

Luego el usuario  $B$  lo que recibe realmente es  $e_B \cdot P_m$ , luego para descifrar el mensaje:

$$P_m = d_B \cdot e_B \cdot P_m$$

Una vez que el usuario  $B$  conoce  $P_m \in E$  tiene que identificar el mensaje  $m$ .

Este método es útil para mandar mensajes entre dos personas y además es seguro.

### 12.5.1. Debilidades de este Sistema Criptográfico

Supongamos que tenemos dos usuarios que van a comunicarse entre sí utilizando este sistema criptográfico, el usuario **A** y el usuario **B**. Supongamos también que hay un tercer usuario, el usuario **C**. Las claves de los usuarios son las siguientes:

Claves privadas	Usuario <b>A</b>	Usuario <b>B</b>	Usuario <b>C</b>
$e$	$e_A$	$e_B$	$e_C$
$d$	$d_A$	$d_B$	$d_C$

Un posible situación que se puede dar es la siguiente:

1. El usuario **A** manda al usuario **B** un mensaje cifrado  $e_A \cdot P_m$ .
2. El usuario **C** intercepta el mensaje, impidiendo que el usuario **B** lo reciba. El usuario **C** asume la identidad del usuario **B** y devuelve al usuario **A** el mensaje recibido, pero en lugar de estar multiplicado por la clave de encriptación del usuario **B**, la cual no conoce, lo manda multiplicado por su clave de encriptación  $e_C$ , luego manda al usuario **A** el mensaje  $e_C \cdot e_A \cdot P_m$ .

3. El usuario **A** recibe el mensaje  $e_C \cdot e_A \cdot P_m$ , creyendo que es la respuesta del usuario **B** y devuelve el mensaje al usuario **B** multiplicado por su clave de descryptación  $d_A \cdot e_C \cdot e_A \cdot P_m = e_C \cdot P_m$ .
4. El usuario **C** vuelve a interceptar la comunicación, impidiendo que el usuario **B** reciba el mensaje. A continuación multiplica el mensaje por su clave de descryptación  $d_C \cdot e_C \cdot P_m = P_m$ , con lo cual el usuario **C** obtiene el mensaje que el usuario **A** había mandado al usuario **B**.
5. El usuario **C** modifica a su conveniencia, si lo estima necesario, el mensaje  $P_m$ . Llamemos  $P'_m$  a dicho mensaje modificado. Entonces el usuario **C** asume la identidad del usuario **A** y manda al usuario **B** el mensaje modificado multiplicado por su clave de encriptación  $e_C \cdot P'_m$ .
6. El usuario **B** recibe el mensaje del usuario **C**, creyendo que es del usuario **A**, y lo devuelve al usuario **A** multiplicado por su clave de encriptación  $e_B \cdot e_C \cdot P'_m$ .
7. El usuario **C** intercepta la comunicación, impidiendo que el mensaje llegue al usuario **A**, y manda al usuario **B** el mensaje multiplicado por su clave de descryptación  $d_C \cdot e_B \cdot e_C \cdot P'_m = e_B \cdot P'_m$ .
8. El usuario **B** recibe el mensaje del usuario **C**, creyendo que es del usuario **A** y lo descrypta utilizando su clave de encriptación. Luego el mensaje que recibe el usuario **B** es  $d_B \cdot e_B \cdot P'_m = P'_m$ .

Podemos ver que el usuario **B** recibe una información a la que ya ha tenido acceso otro usuario, el usuario **C**, pudiendola haber modificado dicho usuario.

Este problema se reduce al problema de la autenticación. Para verificar la autenticidad de un mensaje, o lo que es lo mismo que proviene de quien dice provenir, se utilizan las firmas digitales. Esta sería una buena solución para evitar un ataque del tipo que aquí hemos expuesto.

## 12.6. Equivalencia entre puntos de una Curva Elíptica y Mensajes

Ya hemos visto un sistema criptográfico basado en curvas elípticas, pero quedan pendientes dos detalles:

- Como asignar los puntos de una curva elíptica al mensaje que queremos cifrar.
- Como recuperar un mensaje a partir de los puntos de la curva elíptica.

Supongamos que tenemos una curva elíptica  $E$  definida sobre un cuerpo  $\mathbb{F}_q$ , donde  $q = p^r$ , con  $p$  primo y  $r \in \mathbb{N}$ . Sea  $m$  un texto que queremos encriptar utilizando algún algoritmo basado en curvas elípticas.

### 12.6.1. Como encontrar los puntos de la curva

Los pasos a realizar son los siguientes:

1. Asignaremos a cada letra del mensaje un valor numérico de acuerdo con un alfabeto previamente establecido, como puede ser alguno de los alfabetos mostrados en el apéndice A en la página 127. A cada letra del mensaje le asignaremos un punto de la curva elíptica. En lo sucesivo entenderemos por  $m_i$  al entero que representa a la letra  $i$ -ésima del mensaje  $m$ , y a  $P_{m_i}$  al punto de  $E$  correspondiente a  $m_i$ .
2. Estableceremos un entero  $\kappa \gg 0$  tal que la probabilidad de no poder asignar a  $m_i$  un punto de la curva  $E$  sea de 1 entre  $2^\kappa$ , esto es necesario ya que en  $\mathbb{F}_q$  no todos los elementos poseen raíz cuadrada. En la práctica  $\kappa = 30$ , y, en el peor de los casos con  $\kappa = 50$  debería ser suficiente.

Además de todo esto supondremos lo siguiente:

- $0 \leq m_i < M$  para todo  $m_i$ .
- $q > \kappa \cdot M$ .

Podemos escoger  $M$  como un entero mayor o igual que el número de símbolos que estemos utilizando. Pero hay que tener cuidado, ya que en el sistema criptográfico de Massey-Omura hay que escoger  $M$  de tal forma que verifique:

$$((M - 1) \cdot +\kappa) \cdot \kappa < N(E)$$

$$3. \quad P_{m_i} = (x_i, y_i) \in E.$$

Podemos establecer una correspondencia biyectiva entre el conjunto:

$$\{m_i \cdot \kappa + j\}_{j=1}^{\kappa} \quad (12.1)$$

y un conjunto de puntos de  $\mathbb{F}_q$ . Tenemos que:

$$\mathbb{F}_q = \mathbb{F}_{p^r} = \mathbb{F}_p \times \dots \times \mathbb{F}_p$$

y expresando cada elemento de (12.1) en base  $p$  obtendremos un elemento<sup>4</sup> de  $\mathbb{F}_q$ :

$$a = \sum_{i=0}^{r-1} a_i \cdot p^i = (a_{r-1}, \dots, a_0) \in \mathbb{F}_q$$

llamemos a este elemento de  $\mathbb{F}_q$   $x_i$ , y a continuación calculemos  $y_i$  de la siguiente forma:

$$y_i^2 = x_i^3 + a \cdot x_i + b$$

$y_i$  será la raíz cuadrada de  $y_i$  en  $\mathbb{F}_q$ . Para saber si un elemento de  $\mathbb{F}_q$  posee raíces cuadradas podemos utilizar la “Ley de reciprocidad cuadrática”, y aunque esto no nos da la solución, en el caso de que exista,  $y_i$  de la ecuación  $y_i^2 \equiv y_i \pmod{p}$ , existen algoritmos para determinar la solución, cuando exista. Podemos encontrarnos dos casos:

- La ecuación  $y_i^2 \equiv y_i \pmod{p}$  tiene solución, entonces  $P_{m_i} = (x_i, y_i)$ .
- La ecuación  $y_i^2 \equiv y_i \pmod{p}$  no tiene solución.

Tenemos que  $x_i = m_i \cdot \kappa + j$ , para un cierto  $j \in \{1, \dots, \kappa\}$ , entonces sumamos 1 a  $j$ , con lo que:

$$x_i = m_i \cdot \kappa + j + 1$$

calculamos el nuevo  $y_i^2$  y comprobamos si la ecuación  $y_i^2 \equiv y_i \pmod{p}$  tiene solución. En el caso de no tener solución para ningún valor de  $j$  tendremos que incrementar  $\kappa$  y volver a empezar desde el principio<sup>5</sup>.

Con este método hemos asignado a cada letra  $m_i$  un punto de la curva elíptica  $E$ , pero no hemos encriptado el mensaje. Una vez calculados los puntos de la curva que forman el mensaje hay que proceder a encriptarlos con algún método.

<sup>4</sup>De  $r$  dígitos.

<sup>5</sup>Para todos los  $m_i$  del mensaje.

### 12.6.2. Como obtener el mensaje

Una vez recibido el texto encriptado procederemos a desencriptarlo, entonces obtendremos una serie de puntos de la curva  $E$ , los cuales tenemos que transformar en letras para poder leer el mensaje.

Supongamos que estamos utilizando un alfabeto de  $n$  símbolos y que ya hemos desencriptado el mensaje recibido, obteniendo de esta forma una sucesión de puntos  $\{P_{m_i} = (x_i, y_i)\}_{i=1}^s$  de la curva elíptica  $E$ .

Para obtener la letra correspondiente a cada punto  $P_{m_i}$  procederemos de la siguiente forma:

1. Calculamos  $\kappa^{-1}$  en  $\mathbb{F}_q$ .
2.  $m_i = (x_i - 1) * \kappa^{-1}$  en  $\mathbb{F}_q$ .
  - a) Si  $m_i \in \{0, \dots, n - 1\}$  pasamos al siguiente punto.
  - b) Si  $m_i \notin \{0, \dots, n - 1\}$  tomamos  $x_i = x_i - 1$  y repetimos el calculo hasta que obtengamos que  $m_i \in \{0, \dots, n - 1\}$
3. Consultamos en el alfabeto utilizado que letra se corresponde con  $m_i$ .

## 12.7. El Logaritmo Elíptico

La operación de suma de puntos en una curva elíptica definida en un cuerpo finito  $\mathbb{F}_q$  es una operación costosa de realizar. Si conocemos  $P, Q \in E$  de tal forma que:

$$Q = n \cdot P = P + \dots + P$$

donde  $n$  es un entero no conocido. El cálculo del entero  $n$  no es facil, sobre todo cuando trabajamos en cuerpos de dimensión elevada. Este problema se conoce como el problema del “*logaritmo elíptico*”, definido en el grupo aditivo de una curva elíptica. Es un problema análogo al del “*logaritmo discreto*” en un grupo multiplicativo finito. De esto se deduce que podemos utilizar el logaritmo elíptico como una función de una sola dirección para diseñar sistemas criptográficos de clave pública. La seguridad de un sistema criptográfico basado en funciones de este tipo depende de la complejidad de cálculo del logaritmo elíptico, la cual es mucho mayor que la del logaritmo discreto.

Por ejemplo, el cálculo de un logaritmo elíptico con  $p$  un entero de 30 dígitos requiere aproximadamente unos 3,000 años de un superordenador Cray, mientras que el cálculo de un logaritmo discreto en un cuerpo finito de esa misma dimensión requiere, solamente, un día en ese mismo ordenador.

En la tabla<sup>6</sup> 12.5, en la página 121, podemos ver una comparación entre métodos de curvas elípticas y el RSA. El ordenador utilizado es un ordenador de  $10^9$  FLOPS.

Dígitos	Sistema	Tiempo de ruptura
30	RSA	0'3 Segundos
30	Curvas Elípticas	11 Días
35	RSA	1'2 Segundos
35	Curvas Elípticas	1 Año
40	RSA	6'0 Segundos
40	Curvas Elípticas	$3 \cdot 10^3$ Años
50	RSA	2'0 Minutos
50	Curvas Elípticas	$3 \cdot 10^6$ Años
85	RSA	1 Día
85	Curvas Elípticas	—
100	RSA	28 Días
100	Curvas Elípticas	—
200	RSA	$3'8 \cdot 10^6$ Años
200	Curvas Elípticas	—

Figura 12.5: Comparación entre los métodos RSA y curvas elípticas.

---

<sup>6</sup>Obtenida del libro “Criptografía Digital, Fundamentos y Aplicaciones” de José Pastor Franco y Miguel Angel Sarasa López.

## 12.8. Test de Primalidad con Curvas Elípticas

### Proposición 12.1

Consideremos la curva elíptica  $E$  dada por:

$$y^2 = x^3 + a \cdot x + b$$

y definida sobre  $\mathbb{Z}$ , es decir  $a, b \in \mathbb{Z}$ . Sea  $E_n$  la reducción modulo  $n$  de  $E$ , es decir las soluciones de la curva modulo  $n$ .

Sea  $m$  un entero y  $q$  un primo que divida a  $m$  y que sea mayor que  $(\sqrt[4]{n} + 1)^2$ , entonces si existe un punto  $P \in E$  tal que:

- $m \cdot P = \mathcal{O}$ .
- $\frac{m}{q} \cdot P \neq \mathcal{O}$ .

entonces  $n$  es primo.

### Demostración:

Supongamos que  $n$  no es primo, entonces existirá  $p \in \mathbb{N}$  primo tal que  $p \leq \sqrt{n}$  y que divide a  $n$ .

Sea  $E'$  la curva elíptica  $E$  considerada modulo  $p$  y sea  $m' = N(E')$ , entonces por el teorema de *Hasse* tendremos:

$$m' \leq p + 1 + 2 \cdot \sqrt{p} = (\sqrt{p} + 1)^2 \leq (\sqrt[4]{n} + 1)^2 < q$$

dado que  $q$  es primo y  $m' < q$  tendremos que M.C.D.  $(m', q) = 1$ , luego existirá  $u \in \mathbb{Z}$  tal que:

$$u \cdot q \equiv 1 \pmod{m'}$$

Sea  $P'$  el punto  $P$  considerado modulo  $p$ , por hipótesis tenemos que:

$$m \cdot P = \mathcal{O} \Rightarrow m \cdot P' = \mathcal{O} \Rightarrow u \cdot (m \cdot P') = \mathcal{O} \quad (12.2)$$

$$\frac{m}{q} \cdot P = \mathcal{O} \Rightarrow \frac{m}{q} \cdot P' = \mathcal{O} \quad (12.3)$$

Desarrollando y teniendo en cuenta (12.2) tenemos que:

$$\frac{m}{q} \cdot P' = u \cdot q \cdot \frac{m}{q} \cdot P' = u \cdot m \cdot P' = \mathcal{O} \quad (12.4)$$

lo cual está en contradicción con (12.3). Luego la suposición de que  $n$  no es primo es falsa,  $n$  es primo. ■

### 12.8.1. Descripción práctica del Algoritmo

Sea  $n \in \mathbb{Z}$  un número del que queremos determinar su primalidad. El algoritmo es el siguiente:

1. Seleccionamos, aleatoriamente, tres enteros  $a, x, y$  modulo  $n$ .
2. Calculamos  $b$  de la siguiente forma:

$$b \equiv y^2 - x^3 - a \cdot x \pmod{n}$$

Entonces tenemos que  $P = (x, y)$  es un elemento de  $E$ , donde  $E$  viene dado por:

$$y^2 = x^3 + a \cdot x + b$$

3. Utilizamos el algoritmo de Schoff<sup>7</sup> para encontrar un número  $m$  el cual, si  $n$  es primo, es igual al orden de  $E$  sobre  $\mathbb{F}_n$ . Una vez encontrado  $m$  si no podemos ponerlo de la forma  $m = k \cdot q$ , donde  $k \geq 2$  y es un entero pequeño y  $q$  es un “probable primo”<sup>8</sup> entonces elegimos, de forma aleatoria otros  $a, x, y$  y repetimos el proceso hasta que obtenemos un  $m$  que satisfaga las condiciones anteriores.
4. Calculamos  $m \cdot P$  y  $k \cdot P$ .
  - Si  $m \cdot P \neq \mathcal{O}$  entonces  $n$  es un número compuesto, no es primo, ya que si fuera primo  $m$  sería el orden de  $E$  y entonces  $m \cdot P = \mathcal{O}$  para todo  $P \in E$ .
  - Si  $k \cdot P = \mathcal{O}$ , lo cual no es muy probable, deberemos elegir otros  $a, x, y$  y empezar desde cero.
  - Si  $m \cdot P = \mathcal{O}$  y  $k \cdot P \neq \mathcal{O}$ ,  $k = m/q$ , entonces por la proposición anterior  $n$  es primo, suponiendo que  $q$  sea primo<sup>9</sup>.

Una vez llegado hasta este paso el problema se reduce a probar la primalidad de  $q$ , el cual tiene una magnitud de  $n/2$ , como mucho. Repetimos el mismo proceso, pero sustituyendo  $n$  por  $q$ , reiterando el test de primalidad  $t$  veces.  $t$  no será mayor que  $\log_2 n$ . Entonces habremos obtenido un número  $q_t$  del cual queremos conocer si es o no primo, pero del que sabemos que  $q_{t-1}$  es realmente primo, no probablemente, y repitiendo el mismo razonamiento tendremos que  $q_{t-2}$  es primo y así sucesivamente hasta llegar a  $q_1 = q$ , y al ser  $q$  primo tendremos que  $n$  es primo.

<sup>7</sup>U otro algoritmo para determinar el orden de una curva elíptica.

<sup>8</sup>El número  $q$  satisface algún algoritmo probabilístico de primalidad.

<sup>9</sup>Sólo sabemos que primo con una probabilidad alta.



**Parte VI**  
**Apéndices**



# Apéndice A

## Alfabetos

Para construir un alfabeto, hemos de considerar los símbolos que necesitamos utilizar y etiquetarlos con un número a cada uno. Hay que tener en cuenta que si la cantidad de símbolos utilizada no es un número primo hay que tener cuidado a la hora de dividir, ya que no todos los números tendrán inverso en  $\mathbb{Z}/n$ , por eso hemos incluido un par de alfabetos, el de 29 y 37 símbolos, los cuales contienen una cantidad “prima” de símbolos, de esta manera en algunos ejemplos no tenemos la necesidad de comprobar la existencia de elementos inversos.

### A.1. Alfabeto con 29 símbolos

No consideramos la letra *W* dada su poca frecuencia en el idioma español.

Símbolo	Entero	Símbolo	Entero
“A”	0	“O”	15
“B”	1	“P”	16
“C”	2	“Q”	17
“D”	3	“R”	18
“E”	4	“S”	19
“F”	5	“T”	20
“G”	6	“U”	21
“H”	7	“V”	22
“I”	8	“X”	23
“J”	9	“Y”	24
“K”	10	“Z”	25
“L”	11	“ ”	26
“M”	12	“ , ”	27
“N”	13	“ . ”	28
“Ñ”	14		

Cuadro A.1: Alfabeto de 29 símbolos.

**A.2. Alfabeto con 36 símbolos**

Símbolo	Entero	Símbolo	Entero
“0”	0	“I”	18
“1”	1	“J”	19
“2”	2	“K”	20
“3”	3	“L”	21
“4”	4	“M”	22
“5”	5	“N”	23
“6”	6	“O”	24
“7”	7	“P”	25
“8”	8	“Q”	26
“9”	9	“R”	27
“A”	10	“S”	28
“B”	11	“T”	29
“C”	12	“U”	30
“D”	13	“V”	31
“E”	14	“W”	32
“F”	15	“X”	33
“G”	16	“Y”	34
“H”	17	“Z”	35

Cuadro A.2: Alfabeto de 36 símbolos.

### A.3. Alfabeto con 37 símbolos

No consideramos la letra *W* dada su poca frecuencia en el idioma español.

Símbolo	Entero	Símbolo	Entero
“0”	0	“I”	19
“1”	1	“J”	20
“2”	2	“K”	21
“3”	3	“L”	22
“4”	4	“M”	23
“5”	5	“N”	24
“6”	6	“Ñ”	25
“7”	7	“O”	26
“8”	8	“P”	27
“9”	9	“Q”	28
“ ”	10	“R”	29
“A”	11	“S”	30
“B”	12	“T”	31
“C”	13	“U”	32
“D”	14	“V”	33
“E”	15	“X”	34
“F”	16	“Y”	35
“G”	17	“Z”	36
“H”	18		

Cuadro A.3: Alfabeto de 37 símbolos.

# Apéndice B

## Máquinas de Turing

En las máquinas de Turing tendremos:

- Un conjunto de símbolos:

$$S = \{s_1, s_2, \dots, s_n\}$$

con  $n < \infty$ .

- Un conjunto de estados:

$$E = \{q_0, q_1, \dots, q_r\}$$

con  $r < \infty$ .

- Un conjunto de acciones para la cabeza lectora:

$$L = \{I, D, N\}$$

donde:

1.  $I$  indica que la cabeza lectora se desplazará hacia la izquierda.
2.  $D$  indica que la cabeza lectora se desplazará hacia la derecha.
3.  $N$  indica que la cabeza lectora no se moverá.

Una máquina de Turing consiste en una cinta infinita en la que hay una sucesión infinita de símbolos del conjunto  $S$  y una cabeza lectora que va actuando sobre los elementos de la cinta.

La máquina posee un estado, que es el que determina la acción de la cabeza lectora:

1. Cambia el símbolo.
2. Cambia el estado.
3. Se mueve hacia la izquierda, derecha o no se mueve.
4. La máquina para.

mientras la máquina no pare realiza 1, 2 y 3. El cambio de símbolo y estado depende del estado en el que se encuentre la máquina.

Por ejemplo:

$$S = \{a, b\} \quad E = \{q_0, q_1, q_2\}$$

La forma de actuar de la cabeza lectora dependerá del estado de la máquina y del símbolo que haya leído, luego su forma de actuar vendrá dada por la siguiente tabla:

	a	b
$q_0$	a, $q_1$ ,I	a, $q_0$ ,D
$q_1$	b, $q_2$ ,N	PARAR
$q_2$	PARAR	b, $q_1$ ,N

Cuadro B.1: Tabla de estados y símbolos de una Máquina de Turing.

Donde  $P$  indica que la máquina para, es decir sí la máquina está en el estado  $q_0$  y lee de la cinta el símbolo  $b$  la máquina para y de igual forma para el estado  $q_2$  y el símbolo  $a$ .

Por ejemplo supongamos que el estado inicial de la máquina de Turing es  $q_0$ , podemos ver en la figura B.1 el funcionamiento de la máquina de Turing correspondiente a la tabla B.1.

Diremos que una función es computable Turing sí existe una máquina que la realiza, es decir, existe una máquina de Turing que para cualquier elemento del dominio de la función produce la misma salida que la función.

	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	
...	a	b	b	b	a	a	b	a	b	b	a	b	a	a	...

$q_0$

	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	
...	a	b	b	b	a	a	b	a	b	b	a	b	a	a	...

$q_1$

	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	
...	a	b	b	b	b	a	b	a	b	b	a	b	a	a	...

$q_2$

	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	
...	a	b	b	b	b	a	b	a	b	b	a	b	a	a	...

$q_1$

	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	
...	a	b	b	b	b	a	b	a	b	b	a	b	a	a	...

La máquina de Turing para, ya que lee  $b$  en el estado  $q_1$ .

Figura B.1: Funcionamiento de una Máquina de Turing.



# Apéndice C

## Ejercicios propuestos en clase

### Ejercicio 1

Dados  $m, n \in \mathbb{Z}$  probar que existe  $d \in \mathbb{Z}$  tal que:

$$(m, n) = (d)$$

donde  $(m, n) = \{ \lambda \cdot m + \mu \cdot n \text{ tal que } \lambda, \mu \in \mathbb{Z} \}$  y  $(d)$  es el ideal generado por  $d$ .

### Solución:

$$(m, n) \subseteq (d)$$

Sea  $\lambda \cdot m + \mu \cdot n \in (m, n)$  y  $d = \text{M.C.D.}(m, n)$  entonces tendremos:

$$m = \alpha_1 \cdot d \text{ con } \alpha_1 \in \mathbb{Z}$$

$$n = \alpha_2 \cdot d \text{ con } \alpha_2 \in \mathbb{Z}$$

luego:

$$\lambda \cdot m + \mu \cdot n = \lambda \cdot (\alpha_1 \cdot d) + \mu \cdot (\alpha_2 \cdot d) = (\lambda \cdot \alpha_1 + \mu \cdot \alpha_2) \cdot d \in (d)$$

$$(d) \subseteq (m, n) \text{ con } d = \text{M.C.D.}(m, n)$$

Por definición de  $d$  tenemos:

$$m = \alpha_1 \cdot d \text{ con } \alpha_1 \in \mathbb{Z}$$

$$n = \alpha_2 \cdot d \text{ con } \alpha_2 \in \mathbb{Z}$$

luego  $m, n \in (d)$  y cualquier combinación lineal suya pertenecerá a  $(d)$ , entonces:

$$\lambda \cdot (\alpha_1 \cdot d) + \mu \cdot (\alpha_2 \cdot d) \in (d)$$

según variemos  $\lambda$  y  $\mu$  sobre  $\mathbb{Z}$  iremos obteniendo los elementos de  $(d)$  y todos estos elementos pertenecerán a  $(m, n)$ .

■

### Ejercicio 2

Sea  $n$  un número entero positivo. Demostrar que todo entero  $a$  se puede escribir de la siguiente forma:

$$a = a_0 + a_1 \cdot n + a_2 \cdot n^2 + \dots + a_r \cdot n^r$$

### Solución:

Utilizando el algoritmo de Euclides tendremos:

$$\begin{aligned} a &= q_0 \cdot n + a_0 \\ q_0 &= q_1 \cdot n + a_1 \\ q_1 &= q_2 \cdot n + a_2 \\ &\dots\dots\dots \\ q_{r-1} &= q_r \cdot n + a_r \\ q_r &= q_{r+1} \cdot b + 0 \end{aligned}$$

con  $a_i < n$  para todo  $i$ . En algún momento llegaremos a obtener un resto cero ya que los  $q_i$  van decreciendo y a que estamos operando con números finitos.

Si vamos sustituyendo en la primera expresión  $q_0$  por su valor, después  $q_1$  y así sucesivamente con todos los  $q_i$  llegaremos a una expresión de este tipo:

$$a = a_0 + a_1 \cdot n + a_2 \cdot n^2 + \dots + a_r \cdot n^r$$

■

**Ejercicio 3**

Sean  $P(x), Q(x) \in \mathbb{Z}[x]$  dos polinomios:

$$P(x) = \sum_{i=1}^n a_i \cdot x^i \quad y \quad Q(x) = \sum_{i=1}^m b_i \cdot x^i$$

donde  $a_i, b_i \in \mathbb{Z}^+$ . Determinar el tiempo necesario para calcular  $P(x) \cdot Q(x)$  sabiendo que  $n \geq m$  y que  $a_i, b_i \leq t$ .

**Solución:**

En primer lugar tenemos que:

$$P(x) \cdot Q(x) = H(x) = \sum_{k=0}^{n+m} c_k \cdot x^k \quad \text{donde} \quad c_k = \sum_{i+j=k} a_i \cdot b_j$$

puesto que un polinomio de grado  $m$  tiene  $m + 1$  coeficientes y que  $m \leq n$  tendremos que el cálculo de cada coeficiente  $c_k$  supondrá, como mucho, el cálculo de  $m + 1$  multiplicaciones y  $m$  sumas.

Sabemos, por las hipótesis, que  $a_i$  y  $b_j$  son menores o iguales que un cierto  $t$  para todos los  $i, j$ , luego los números que estaremos sumando serán menores o iguales que  $t^2$ . Y como tenemos que sumar  $m$  de estos números tendremos que el mayor número que podemos obtener de estas operaciones es  $m \cdot t^2$ .

Sabemos que el número de bits necesarios para representar  $t$  será:

$$\lceil \log_2 t \rceil + 1$$

y de igual modo el número de bits necesarios para representar  $m \cdot t^2$  será:

$$\lceil \log_2(m \cdot t^2) \rceil + 1$$

De todo esto se deduce que el número de operaciones para multiplicar  $m + 1$  números menores que  $t$  es:

$$(m + 1) \cdot (\lceil \log_2 t \rceil + 1)^2$$

y que el número de operaciones necesarias para sumar  $m$  números menores o iguales que  $m \cdot t^2$  son:

$$m \cdot (\lceil \log_2(m \cdot t^2) \rceil + 1)^2$$

por lo tanto el número de operaciones necesario para el cálculo de cada coeficiente será:

$$(m + 1) \cdot ([\log_2 t] + 1)^2 + m \cdot ([\log_2(m \cdot t^2)] + 1)^2$$

Como el polinomio  $H(x) = P(x) \cdot Q(x)$  es de grado  $n + m$  tiene  $n + m + 1$  coeficientes, luego el número de operaciones, a nivel de bits, necesario para calcular dicho polinomio será:

$$(n + m + 1) \cdot \left( (m + 1) \cdot ([\log_2 t] + 1)^2 + m \cdot ([\log_2(m \cdot t^2)] + 1)^2 \right)$$

■

#### Ejercicio 4

Sean  $n, m \in \mathbb{Z}^+$  con  $n \geq m$ . Determinar el tiempo necesario para calcular:

$$\binom{n}{m} = \frac{n!}{m! \cdot (n - m)!} = \frac{n \cdot (n - 1) \dots (n - m + 1)}{m!}$$

#### Solución:

Tenemos que realizar  $m - 1$  multiplicaciones y luego  $m - 1$  divisiones.

Tenemos que  $n \cdot (n - 1) \cdot \dots \cdot (n - m + 1) < n^m = n \cdot n^{m-1}$ . Como el número de bits de  $n$  es  $[\log_2 n] + 1$  y teniendo en cuenta que el número de bits de un producto es menor o igual que la suma de los bits de los números multiplicados tendremos que el número de bits de  $n^m$  es menor o igual que  $m \cdot ([\log_2 n] + 1)$ , y como para calcular  $n^m$  tenemos que realizar  $m - 1$  multiplicaciones entonces tendremos que el número de operaciones, a nivel de bits, que tenemos que realizar es:

$$(m - 1) \cdot m \cdot ([\log_2 n] + 1)^2$$

y como también tenemos que realizar  $m - 1$  divisiones el número total de operaciones será:

$$2 \cdot (m - 1) \cdot m ([\log_2 n] + 1)^2$$

■

**Ejercicio 5***Sea:*

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

con  $a_{ij} \in \mathbb{Z}/n$ .Probar que existe  $A^{-1} \iff |A| \in (\mathbb{Z}/n)^*$ **Solución:** $\Rightarrow$  |  $A$  es invertible, entonces existe  $A^{-1}$ :

$$A \cdot A^{-1} = I$$

de esta identidad tenemos:

$$|A \cdot A^{-1}| = |I| = 1$$

luego:

$$|A \cdot A^{-1}| = |A| \cdot |A^{-1}| = 1$$

luego  $|A| \in (\mathbb{Z}/n)^*$ . $\Leftarrow$  |  $|A| = (a_{11} \cdot a_{22}) - (a_{12} \cdot a_{21}) \in (\mathbb{Z}/n)^*$ . Definamos la siguiente matriz:

$$\frac{1}{|A|} \cdot B = \frac{1}{|A|} \cdot \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

Esta matriz es la inversa de  $A$ :

$$\begin{aligned} A \cdot \frac{1}{|A|} \cdot B &= \frac{1}{|A|} \cdot \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} = \\ &= \frac{1}{|A|} \cdot \begin{pmatrix} a_{11} \cdot a_{22} - a_{12} \cdot a_{21} & a_{11} \cdot (-a_{12}) + a_{11} \cdot a_{12} \\ a_{21} \cdot a_{22} - a_{22} \cdot a_{21} & -a_{21} \cdot a_{12} + a_{11} \cdot a_{22} \end{pmatrix} = \\ &= \frac{1}{|A|} \cdot \begin{pmatrix} |A| & 0 \\ 0 & |A| \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{aligned}$$

De forma análoga se comprueba para  $1/(|A|) \cdot B \cdot A = I$ . Luego existe la inversa.

■

**Ejercicio 6**

Sea  $n = 26$ ,  $E = \mathbb{Z}/26 \times \mathbb{Z}/26$  y con la matriz de encriptación:

$$A = \begin{pmatrix} 2 & 3 \\ 7 & 8 \end{pmatrix}$$

descifrar “FWMDIQ”.

**Solución:**

Primero tenemos que calcular la inversa de la matriz en  $\mathbb{Z}/26$ .

$$A^{-1} = \frac{1}{|A|} \cdot \begin{pmatrix} 8 & -3 \\ -7 & 2 \end{pmatrix}$$

Como  $|A| = 21$ , por el algoritmo de Euclides resolvemos

$$a \cdot 21 + 26 \cdot b = 1$$

y obtenemos que  $a = 5$ , luego  $21^{-1} = 5 \pmod{26}$ . Luego tendremos que:

$$A^{-1} = 5 \cdot \begin{pmatrix} 8 & -3 \\ -7 & 2 \end{pmatrix} = \begin{pmatrix} 14 & 11 \\ 17 & 10 \end{pmatrix} \pmod{26}$$

Como  $n = 26$  el alfabeto que estamos utilizando  $A = 0, B = 1, \dots, Z = 25$ . Luego:

$$\text{FWMDIQ} = (5, 22, 12, 3, 8, 16)$$

Para descifrar el mensaje:

$$\begin{aligned} C &= \begin{pmatrix} 14 & 11 \\ 17 & 10 \end{pmatrix} \cdot \begin{pmatrix} 5 & 12 & 8 \\ 22 & 3 & 16 \end{pmatrix} = \begin{pmatrix} 312 & 201 & 288 \\ 305 & 234 & 296 \end{pmatrix} = \\ &= \begin{pmatrix} 0 & 19 & 2 \\ 19 & 0 & 10 \end{pmatrix} \pmod{26} \end{aligned}$$

Consultando el alfabeto utilizado tendremos que:

$$(0, 19, 19, 0, 2, 10) = \text{ATTACK}$$

■

**Ejercicio 7**

Dada la siguiente curva elíptica:

$$y'^2 + y' = x^3 - x$$

la cual se sabe que tiene 727 puntos en  $\mathbb{F}_{751}$  realizar de forma efectiva el algoritmo de Massey-Omura.

**Solución:**

Aunque podemos aplicar el algoritmo a la curva dada originalmente vamos a transformarla para tenerla en la forma en que hemos utilizado en los ejemplos. Haciendo el cambio de variable  $y' = y - 376$  tenemos:

$$y^2 - 751y + 141,000 = x^3 - x$$

como la característica de  $\mathbb{F}_{751}$  es 751 tendremos que la ecuación de la curva será:

$$y^2 = x^3 - x - 563$$

$N(E) = 727$ . Vamos a utilizar el alfabeto de 36 símbolos<sup>1</sup>. Para ver si un determinado elemento de  $\mathbb{F}_{751}$  posee raíz cuadrada utilizamos el programa Mathematica.

Claves privadas	Usuario <b>A</b>	Usuario <b>B</b>
$e$	20	91
$d$	618	8

Fijamos  $\kappa = 20$  y supongamos que el usuario **A** quiere transmitir la palabra “HOLA” al usuario **B**:

- Lo primero es encontrar los puntos de la curva asociados a la palabra:

- $H = 17$ .  $x_1 = 17 \cdot 20 + 1 = 341$ . Calculamos  $y_1^2$  en  $\mathbb{F}_{751}$ :

$$y_1^2 = 341^3 - 341 - 563 = 370$$

La raíz cuadrada, una de ellas, en  $\mathbb{F}_{751}$  de 370 es 362.

Tenemos que a la letra “H” le asignaremos el siguiente punto:

$$P_H = (341, 362) \in E$$

---

<sup>1</sup>Página 129.

- O = 24.  $x_2 = 24 \cdot 20 + 1 = 481$ . Calculamos  $y_2^2$  en  $\mathbb{F}_{751}$ :

$$y_2^2 = 481^3 - 481 - 563 = 417$$

Como 417 no tiene raíz cuadrada en  $\mathbb{F}_{751}$  incrementamos en 1  $x_3$  hasta encontrar un valor que tenga raíz cuadrada.

Tenemos que a la letra “O” le asignaremos el siguiente punto:

$$P_O = (484, 590) \in E$$

- L = 21.  $x_3 = 21 \cdot 20 + 1 = 421$ . Calculamos  $y_3^2$  en  $\mathbb{F}_{751}$ :

$$y_3^2 = 421^3 - 421 - 563 = 370$$

La raíz cuadrada, una de ellas, en  $\mathbb{F}_{751}$  de 370 es 362.

Tenemos que a la letra “L” le asignaremos el siguiente punto:

$$P_L = (421, 362) \in E$$

- A = 10.  $x_4 = 10 \cdot 20 + 1 = 201$ . Calculamos  $y_4^2$  en  $\mathbb{F}_{751}$ :

$$y_4^2 = 201^3 - 201 - 563 = 25$$

La raíz cuadrada, una de ellas, en  $\mathbb{F}_{751}$  de 25 es 5.

Tenemos que a la letra “A” le asignaremos el siguiente punto:

$$P_A = (201, 5) \in E$$

2. El mensaje, sin encriptar, que el usuario **A** quiere mandar es:

$$(341, 362)(484, 590)(421, 362)(201, 5)$$

Como el procedimiento es el mismo para todos los puntos unicamente lo haremos para el punto  $(484, 362)$ , que representa a la letra “O”.

Los múltiplos de los puntos de la curva elíptica los hemos calculado utilizando el algoritmo de “izquierda a derecha”<sup>2</sup>, y para calcular la

---

<sup>2</sup>Apartado 12.4.1 en la página 113.

suma de dos puntos de la curva elíptica hemos utilizado el algoritmo de cálculo en “*coordenadas proyectivas*”<sup>3</sup>, utilizando el programa Mathematica<sup>4</sup>.

Recordemos que:

$$n \cdot P = P + \dots + P$$

donde  $P \in E$  y  $n \in \mathbb{F}_q$ , cuerpo sobre el que está definida la curva elíptica.

El procedimiento será el siguiente:

- a) El usuario **A** manda al usuario **B** el punto multiplicado por su clave de encriptación,  $e_A = 20$ :

$$20 \cdot (484, 590) = (317, 181)$$

- b) El usuario **B** manda al usuario **A** el mensaje recibido, pero multiplicado por su clave de encriptación,  $e_B = 91$ :

$$91 \cdot (317, 181) = (311, 640)$$

- c) El usuario **A** devuelve al usuario **B** el mensaje multiplicado por su clave de desencriptación,  $d_A = 618$ :

$$618 \cdot (311, 640) = (663, 494)$$

- d) El usuario **B** recibe  $(663, 494)$ , para obtener el punto original sólo tiene que multiplicar por su clave de desencriptación,  $d_B = 8$ :

$$8 \cdot (663, 494) = (484, 590)$$

3. Ahora el usuario **B** tiene un punto de la curva elíptica  $E$ . Para recuperar la letra del mensaje utilizaremos la siguiente fórmula, teniendo en cuenta que estamos utilizando un alfabeto con 36 símbolos:

$$m_i = \frac{x_i - 1}{\kappa} \bmod 751 \in \{0, \dots, 35\}$$

<sup>3</sup>Apartado 12.3.2 en la página 111.

<sup>4</sup>En la página 113 se puede ver un ejemplo práctico de como se ha calculado el primer paso del algoritmo.

Lo primero es calcular  $\kappa^{-1}$  en  $\mathbb{F}_{751}$ :

$$20^{-1} = 338 \text{ en } \mathbb{F}_{751}$$

$m_2 = (x_2 - 1) * 338 = (484 - 1) * 338 = 287$ , como  $287 \notin \{0, \dots, 35\}$  entonces tomamos  $x_2$  como  $x_2 - 1$  y repetimos el calculo hasta que obtengamos un elemento de  $\{0, \dots, 35\}$ :

$$m_2 = ((x_2 - 1) - 1) * 338 = ((481 - 1) - 1) * 338 = 24$$

Consultando el alfabeto que hemos utilizado tendremos:

$$m_2 = 24 \Rightarrow m_2 = \text{O}$$

Luego la palabra transmitida por el usuario **A** es “O”.



# Bibliografía

- [1] Apuntes de clase.
- [2] “*Codes and Cryptography*”. Dominic Welsh. Oxford Science Publications.
- [3] “*A Course in Number Theory and Cryptography*”, Second Edition. Neal Koblitz. Springer.
- [4] “*Criptografía Digital, Fundamentos y Aplicaciones*”. José Pastor Franco y Miguel Angel Sarasa López. Colección textos docentes Prensas Universitarias de Zaragoza.
- [5] “*Criptografía y Seguridad en Computadores*”, Segunda Edición. Manuel José Lucena López.
- [6] Artículo “*Zoología de los números*” de Maurice Mashall. Publicado en la revista “*Mundo Científico*” 161 volumen 15.
- [7] Artículo “*Los números primos*” de Henri Cohen. Publicado en la revista “*Mundo Científico*” 161 volumen 15.