1 Aritmética Modular y Criptografía

1.1 Glosario

Pretendemos dar las bases mínimas para poder mostrar aplicaciones a la Criptografía de la Aritmética modular. Para ello introducimos brevemente cierta terminología necesaria.

Criptografía: la ciencia o arte de mantener mensajes seguros, para ello el mensaje se cifra (encripta).

Criptoanálisis: la ciencia o arte de romper códigos cifrados (criptografiados).

Interlocutores: aquellas personas que utilizan la criptografía para enviarse mensajes en forma confidencial. Alternativamente juegan el papel de transmisor o receptor de mensajes.

espía: o criptoanalista o adversario u oponente, aquella persona u organización que pretende recuperar cierto mensaje a partir del texto encriptado.

texto llano: mensaje sin cifrar, tal como es emitido por los interlocutores. Hemos optado por traducir la expresión inglesa "plain text" como texto llano.

texto cifrado: o encriptado, mensaje después de la operación de encriptar.

clave simétrica: sistema de encriptar que utiliza una clave k solo conocida por aquellos interlocutores autorizados. Estos deben ponerse de acuerdo en la clave previamente a intentar comunicarse.

clave pública: o asimétrica, sistema de encriptar que utiliza una clave pública e_B que permite que un usuario **A** envíe un mensaje a otro **B** de modo que sólo este último pueda leerlo. Para ello **B** utiliza una clave secreta d_B (solo conocida por él). Se supone que el conocimiento de la clave pública e_B no habilita a un espía a conocer la clave secreta d_B ni el mensaje por algún otro método que pudiera existir y que no utilice d_B. Las claves públicas se listan en un directorio similar a una guía telefónica.

1.2 Preliminares

Para encriptar un mensaje necesitamos previamente representarlo en forma numérica. Esto no es un gran problema dado que las computadoras de hecho es como representan los caracteres y letras. Así la letra 'a' corresponde en código ASCII al número

decimal 97, la letra 'o' se representa como 111, y el número 2, también en ASCII, se representa como 50. Por ejemplo para codificar 'onda2' en ASCII escribimos los numeros: 111 110 100 97 50
La encriptación, más propiamente llamada cifrado, consiste en realizar transformaciones sobre el texto representado en forma numérica de modo que su apariencia difiera del texto original tanto que sea irreconocible. Algunas propiedades que necesariamente deben tener estas transformaciones son:

- deben ser invertibles, para a partir del texto cifrado ser capaces de recuperar el texto original,
- tanto el proceso de encriptar como el de desencriptar deben ser fáciles de realizar **si uno posee las claves necesarias para ello**,
- el proceso de desencriptar debe ser prácticamente imposible de realizar si no se posee la clave para hacerlo.

Podemos realizar un símil con la apertura de una caja de seguridad; si se conoce la combinación se tardará algunos segundos en abrir la caja. Si no es así, se deberá probar tanteando todos los casos posibles, y salvo una gran casualidad se tardará un tiempo tan grande en abrir la caja que tornará esta tarea en impracticable. En el último punto tratado se consideró un método para abrir la caja que consiste en probar todos los casos posibles. Es lo que se llama un ataque de 'fuerza bruta'. Lo mismo es válido para la Criptografía, un espía puede siempre intentar descifrar un mensaje probando todos los casos posibles.

Un método de encriptar se considera seguro si el mejor método aplicado para intentar descifrar un mensaje es equivalente al ataque por fuerza bruta.
Con el aumento de la velocidad de los ordenadores se logra que el ataque por fuerza bruta sea cada vez más eficaz. Si un computador puede probar 1 millón de claves por segundo, la siguiente tabla da una idea del valor esperado que tarda en hallar la clave correcta dependiendo del largo de la clave medido en bytes Se recuerda que un byte esta formado por 8 bits. Como cada bit puede valer 0 o 1, los posibles valores de un byte son 256 en total, para dos bytes son $2^{16} = 65536$, para 4 bytes son $2^{32} = 4294967296 \approx 10^{9,63}$. Para 8 bytes tenemos una clave del orden de 10^{19}. El tiempo de búsqueda en un ataque por fuerza bruta crece exponencialmente con

Largo clave	4 bytes	6 bytes	8 bytes
Letras minúsculas (26)	.5 seg	5 min	2.4 días
Caracteres alfanuméricos (62)	15 seg	16 horas	6.9 años
Caracteres ASCII de 8 bits	1.2 horas	8.9 años	580000 años

Tabla 1: Tiempos de búsqueda sistemática (a un millón de tentativas por segundo)

el largo de la clave. Hoy en día, suponiendo que se posean los recursos más potentes en computación, y que el ataque por fuerza bruta sea el más eficiente se consideran seguras claves de una longitud de 16 bytes, unos 128 bits. Cuando decimos esto, nos referimos a sistemas de encriptado donde la clave de encriptar y la de desencriptar son ambas secretas. Tales sistemas se llaman de clave simétrica o privada. La característica de tales sistemas es que conociendo la clave para encriptar, la de desencriptar o bien coincide con la de encriptar, o bien el cálculo de la clave de desencriptar a partir de la de encriptar es sencillo. Para los sistemas de encriptar de clave p'ublica las claves deben ser mucho más largas para tener el mismo nivel de seguridad.

Lo que se usa al establecer una comunicación encriptada con clave simétrica, entre dos interlocutores es generar una clave **al azar** para esa comunicación concreta. El problema es que los interlocutores deben ponerse de acuerdo en la clave sin revelarla a un posible espía!

1.3 Sistemas simétricos

Daremos a continuación un ejemplo simple de sistema de encriptar simétrico. Su uso, en la forma más ingenua, se remonta a la antigua Roma, y se supone que era utilizado por Cesar. Se representa cada letra con un numero de 0 a 26 (contamos la letra "ñ"), así a la "a" le corresponde el 0, a la "b" el 1, a la "c" el 2 etc., hasta llegar a la "z" a la que corresponde el 26. El sistema consiste en desplazar un numero fijo k los valores así obtenidos, sumando módulo 27. Si por ejemplo $k = 3$, entonces a la "a" le corresponde el $0 + 3 = 3$ que es la letra "d", si nos pasamos de 26, volvemos a empezar por 0. Por ejemplo, a la "y" le corresponde

$25 + 3 = 28 = 1 \, mod \, 27$, el 1 está asociado con la letra "b". El mensaje "nos ganaron otra vez" se vería como "prv jdpdurp rwud yhc".

Este sencillo método puede que resultara útil para encriptar en la época de la Antigua Roma, pero es totalmente débil, en particular no resiste una búsqueda por fuerza bruta ni siquiera a mano, dado que los posibles valores de k no superan el numero 26. En efecto, al sumar a n el valor 27 se vuelve a obtener n módulo 27. Sabiendo que se usó el método Cesar para encriptar, sólo hay que probar con 26 casos posibles.

Dada la forma usada al encriptar, otro tipo de ataque es posible: en el Castellano, en Inglés y en cualquier otro idioma, no todos los caracteres aparecen con la misma frecuencia. La letra "e" es la más frecuente tanto en Castellano como en Inglés, en Castellano le sigue la "a". Y por supuesto, el carácter "espacio en blanco" que podemos representar con β es aun más frecuente. En el texto cifrado "prv jdpdurp rwud yhc" vemos que se repite mucho la letra "d". Podría sospecharse que es la "e", en cuyo caso k sería 26 (para que al sumar el valor que corresponde a la "e", o sea el 4, me de el valor que corresponde a la "d", o sea el 3: $4 + 26 = 30 = 3 \, mod \, 27$). Al hacer esta suposición, obtenemos descifrando con $k = 26$ (tenemos que deshacer el proceso, o sea restar 26 que es lo mismo que sumar 1 módulo 27) el texto "qsw keqevsq sxve zid" que no tiene sentido, y por lo tanto decidimos probar sustituyendo "d" por la otra letra que aparece más frecuentemente, la "a". Eso da $k = 3$ que sabemos que es el valor correcto.

El ataque que hace uso de la frecuencia con que aparecen los caracteres se llama Análisis Frequencial. Para prevenir este ataque puede usarse lo siguiente: en lugar de encriptar de a un caracter, lo hacemos de a parejas de caracteres que se llaman digrafos, o de a ternas de caracteres que se llaman trigrafos, o en general en bloques de l caracteres.

Supongamos que nuestro alfabeto, va a consistir de las letras minúsculas, seguidas del espacio en blanco β, los números de 0 a 9, los signos de puntuación "!", "." y "?". Esto hace 40 caracteres. que pueden representarse con números de 0 a 39 módulo 40 y al trabajar con digrafos, a cada par de caracteres le asociamos un número entre 0 y $1599 = 16^2 - 1$ como sigue: "aa" $\mapsto 0$, "ab" $\mapsto 1$, "ac" $\mapsto 2$, ..., "cβ $\mapsto 2 \cdot 40 + 27 = 107$, etc.. El mayor valor corresponde a "??" según el orden que hemos elegido nosotros para los símbolos. Esto da $39 \cdot 40 + 39 = 1599$. En este caso

aumentamos nuestro número de claves que ahora son prácticamente $40 \cdot 40 = 1600$. Supongamos que encriptamos nuestro mensaje, "nos ganaron otra vez" de este modo. Primero cambiamos en el mensaje los espacios en blanco por β para que sea más claro lo que queremos hacer. Obtenemos "nosβganaronβotraβvez". Los digrafos son "no", "sβ", "ga", "na", "ro", "nβ", "ot", "ra", "βv", "ez".

Si por ejemplo tomamos como clave a $k = 255$ y aplicamos el método Cesar con el módulo N igual a 1600, el método de encriptar es ahora

- llevar los digrafos a números x (en caso de que el número de caracteres sea impar, agregar un carácter de relleno al final, previamente convenido entre los interlocutores).
- sumar a cada número x la clave k: $y = x + k \, mod \, N$, en nuestro caso hacer $y = x + 255 \, mod \, 1600$.
- reescribir el número y como digrafo.

Por ejemplo, a "no" le corresponde $13 \cdot 40 + 14 = 534$, al sumarle 255 nos queda 789. Para saber qué digrafo le corresponde lo escribimos en base 40 que nos da $19 \cdot 40 + 29$. La pareja $(19, 29)$ se asocia al digrafo "s1" pues a la "s" le corresponde el 19 y al "1" le corresponde el 29. (A la "z" le corresponde el 26, luego viene "β" con el 27, el "0" con el 28, y a continuación el "1" con el 29). Ejercicio: codificar "nosβganaronβotraβvez". Si bien esto es más seguro, el número de claves usadas es insuficiente para los recursos computacionales de que hoy se dispone. Tampoco resiste el análisis frecuencial si el texto es relativamente largo. Encriptando de a digrafos aparecerá frecuentemente "eβ", la "e" seguida de un espacio en blanco. Otros digrafos, como "zy", es prácticamente imposibles que aparezcan.

En la práctica se codifica tomando bloques de l caracteres donde $l \geq 8$. Cada carácter ocupa un byte (= 8 bits). Cada bloque se traduce en un número. Dado el tamaño de los bloques, los cálculos con los números así obtenidos suelen requerir software especial, sea para poder representar los números, sea para acelerar los cálculos. Si trabajamos con todos los caracteres ASCII debemos considerar 128 valores posibles por cada byte, tomando por ejemplo 8 bytes (un número considerado excesivamente pequeño) esto nos da números entre 0 y $N = 128^8 \approx 10^{17}$. Ya para 16 bytes estamos en el orden de $N = 10^{33}$. El método Cesar no se utiliza, ya que es completamente inseguro, no importa lo grande que se tome el bloque.

Una variante un poco más segura es la siguiente: A cada número x se le hace corresponder $y = ax + b \, mod \, N$. Para que la decodificación no sea ambigua, el valor de a debe ser un invertible módulo N. La condición necesaria y suficiente para que a sea invertible en $\mathbb{Z}_N$ es que $MCD(a, N) = 1$, entonces el número de invertibles módulo N es igual a la cantidad de naturales a tales que $a < N$ que cumplen $MCD(a, N) = 1$. Si por ejemplo N es primo, todos los números entre 1 y $N - 1$ cumplen que $MCD(a, N) = 1$ y por lo tanto en este caso hay $N - 1$ valores posibles para a. En general, el número de posibilidades para a tales que $1 \leq a < N$ que cumplen $MCD(a, N) = 1$ está dado por la función de Euler de N, $\varphi(N)$. Si la descomposición factorial de N es

$$N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$$

donde $p_1, p_2, \ldots, p_k$ son los factores primos de N, entonces $\varphi(N)$ vale

$$\varphi(N) = N(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k})$$

Si por ejemplo $N = 1600 = 2^6 \cdot 5^2$ entonces

$$\varphi(N) = \varphi(1600) = 1600(1 - \frac{1}{2})(1 - \frac{1}{5}) = \frac{1600 \cdot 4}{2 \cdot 5} = 640$$

El número b puede en principio tomar cualquier valor módulo N. El número de claves queda entonces igual a todos los valores posibles que la pareja (a, b) pueda tomar, o sea $\varphi(N) \cdot N$. En el ejemplo con $N = 1600$ esto da $640 \cdot 1600 = 1024000$ claves posibles. Este número es francamente insuficiente.

El método descripto aquí para encriptar se conoce como sistema afín.

Ejercicio: Para el caso anterior, con $N = 40^2 = 1600$ y la clave $k = (a = 101, b = 255)$ encriptar usando el sistema afín
"nosβganaronβotraβvez!βhastaβcuandoβparaguas?".

Ejercicio: Trabajando con trigrafos, y con un alfabeto que al igual que antes tiene 40 símbolos hallar el número de claves posibles encriptando en sistema afín, $x \mapsto y = ax + b$.

Ejercicio: Encriptar por trigrafos con $a = 23003$, $b = 5718$ el texto: "Volverán las oscuras golondrinas de tu balcón sus nidos a colgar". No preocupase por los acentos (como en "Volverán").

1.4 Clave Pública

Los algoritmos de clave pública se desarrollan en un principio a partir de los trabajos de R. Rivest, A. Shamir, L. Adleman, W. Diffie, M. Hellman, R. Merkle y ElGamal. Permiten un alto grado de seguridad unido a facilidad de manejo, pues se elimina el problema del intercambio de claves entre *cada par* de usuarios que desean comunicarse en forma segura. Permite que cada usuario haga público el modo de encriptarle mensajes para que él los entienda, sin que a partir de esos datos se pueda saber cómo desencriptar. A partir del hecho de que sólo el usuario sabe cómo desencriptar es posible implementar un método de firma electrónica.
Los sistemas de clave pública conocidos se basan para lograr esto en algún problema matemático considerado difícil. El que presentamos aquí, conocido como RSA por quienes lo propusieron (Rivest, Shamir y Adleman) se basa en la dificultad de hallar los factores primos de un número N que se toma como el producto de dos primos muy grandes p, q, $p \neq q$; $N = pq$. Este sistema es hoy en día el más popular de los usados de clave pública. Para factorizar N,si p y q son del mismo orden de magnitud, la búsqueda deberá seguir hasta aproximadamente $\sqrt{N}$, pues como $N = pq$ y p y q son del mismo orden, no diferirán demasiado **en orden de magnitud** de $\sqrt{N}$. Esto da que la cantidad de números a intentar son de un orden parecido a $\sqrt{N}/3$, esta estimacion toma en cuenta que no vamos a dividir por numeros pares, ni aquellos que son multiplos de 3 o 5. En base 2, al igual que en base 10, existen reglas simples que permiten descartar estos numeros. Lo ideal sería dividir solo por numeros primos, pero averiguar si cierto numero es primo o no es algo costoso computacionalmente. Si por ejemplo $N = 38807$ pruebe el lector "a mano" de hallar p y q. En definitiva, el número de operaciones (divisiones esencialmente) involucradas para descomponer N como pq es del orden de $\sqrt{N}$ si usamos el algoritmo clásico aprendido en la escuela.

Existen algoritmos mucho más veloces, pero no se conoce (y se conjetura que no lo hay) ningún algoritmo realmente veloz para descomponer un número en factores primos, a menos que el número en sí tenga alguna forma muy especial.

Un entero n escrito en base b se escribe en la forma

$$(*) \quad n = (d_{k-1}d_{k-2}\cdots d_1 d_0)_b$$

donde los números d_j son símbolos que representan los dígitos del 0 al $b-1$ y la

expresión $(*)$ representa el número

$$n = d_{k-1} \times b^{k-1} + d_{k-2} \times b^{k-2} + \cdots + d_1 \times b + d_0.$$

Así en la usual base $b = 10$ se usan los símbolos $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$, y la expresión $(9324)_{10}$ se escribe simplemente 9324 ya que se sobreentiende la base 10, y representa

$$9 \times 1000 + 3 \times 100 + 2 \times 10 + 4 = 9 \times 10^3 + 3 \times 10^2 + 2 \times 10 + 4$$

Si el primer dígito d_{k-1} es distinto de cero, decimos que n es un número de k dígitos en la base b. Al variar la base, un número puede necesitar distinta cantidad de dígitos para ser representado. Por ejemplo, $(27)_{10} = (102)_5 = (11011)_2$ ya que $(27)_{10} = 2 \times 10 + 7 = 27$ y también $(102)_5 = 1 \times 5^2 + 0 \times 5 + 2 = 25 + 2 = 27$ y $(11011)_2 = 1{\times}2^4{+}1{\times}2^3{+}0{\times}2^2{+}1{\times}2{+}1 = 16{+}8{+}2{+}1 = 27$. Aquí estamos usando los símbolos $0, 1, 2, 3, 4$ en la base 5 y $0, 1$ en la base 2. Si $n = (d_{k-1}d_{k-2}\cdots d_1d_0)_b$, y $d_{k-1} \neq 0$, entonces se cumple que $b^{k-1} \leq n < b^k$. Tomando logaritmos en base b se tiene entonces que $k - 1 \leq \log_b n < k$. Tomando la parte entera del logaritmo y sumándole 1 tenemos entonces que $k = [\log_b n] + 1 =$ (número de dígitos de n en la base b). Cada computadora tiene un tiempo interno, propio del modelo, para realizar las operaciones más sencillas que para nosotros van a ser sumar un dígito con otro, vale decir, dado que los ordenadores trabajan en base 2, sumar $0 + 0$, $0{+}1$, $1{+}0$, $1{+}1$. Nosotros asignaremos un valor de 1 al tiempo que insume una tal operación. Dependiendo del modelo de máquina, este tiempo podrá ser mayor o menor. Al sumar dos números $n = (d_{k-1}d_{k-2}\cdots d_1d_0)_2$ y $m = (c_{l-1}c_{l-2}\cdots c_1c_0)_2$, suponiendo para fijar ideas que $l \leq k$, tenemos que realizar a lo sumo k sumas de dígitos. En este caso vemos entonces que el tiempo insumido por la suma es k sumas de dígitos. Diremos que el costo de esta operación es de $\log_b(\max\{n, m\}) = k$.
Análogamente el costo de la resta $n-m$ es $\log_b(\max\{n, m\})$. Para la multiplicación de n por m se puede ver que por el método escolar usual de multiplicación, su costo es de no más de $l(k + l) \leq 2kl \leq 2k^2$.

1.4.1 Notación O mayúscula

Supongamos que tenemos dos funciones $f(n)$ y $g(n)$, $n \in \mathbb{N}$, $f, g : \mathbb{N} \to \mathbb{R}^+$ (f y g toman valores *reales positivos*), diremos que $f(n) = O(g(n))$, si existe una

constante $K > 0$ tal que $\forall n : f(n) \leq Kg(n)$. Por ejemplo, si $f(n) = 6n^3 - 3n^2 + 5n$ y $g(n) = n^3$ entonces $f(n) = O(g(n)) = O(n^3)$. Aplicado a los costos de la suma podemos escribir que es un $O(k) = O(\log_b(\max\{n, m\}))$. Del mismo modo el costo de la multiplicación es $O(k^2) = O(\log_b(\max\{n, m\})^2)$. Puede verse que la división entera aprendida en la escuela insume un tiempo para ser ejecutada que es del orden de kl operaciones de dígito, tanto para hallar el cociente como para hallar el resto, donde k es la longitud del dividendo y l la del divisor. Podemos escribir entonces que su costo es un $O(kl)$, y si suponemos que $l \leq k$ podemos también escribir que su costo es un $O(k^2)$. La notación $f = O(g)$ debe entonces verse más como una estimación por exceso, que como una verdadera igualdad.
Con la notación introducida puede verse que el número de operaciones que son necesarias para, **aplicando el algoritmo clásico para descomponer N en factores primos**, es

$$\sqrt{N} \times (\text{costo de una división}) = O(\sqrt{N}(\log(N))^2)$$

Si la longitud de N es $k \approx \log_2(N)$ esto queda

$$O(\sqrt{N}\log(N)) = O(2^{k/2} \cdot k^2)$$

Es entonces exponencial en la longitud k de N. Para que un algoritmo se considere eficiente, su complejidad debe ser a lo sumo polinomial en la longitud de N. La conjetura es que tal algoritmo **no existe** para el problema de la factorización. Hablando en forma ligera, el sistema RSA de encriptar, basa su seguridad en que un espía no puede desencriptar un mensaje, a menos que sepa factorizar N, cosa que no puede hacer en un tiempo razonable.

1.4.2 Sistema RSA

A continuación damos el esquema de cómo funciona el método de encriptar RSA. En primer lugar, se eligen números primos p y q **al azar**. Se recomienda que sean de más de 200 cifras decimales, de igual o muy similar número de bits pero bastante distintos entre sí. Esto es, $p, q \sim 10^{200}$ y $|p - q| \sim 10^{90}$. Si su diferencia es demasiado pequeña un espía puede llegar a encontrarlos a partir de un método para factorizar conocido como Método de Fermat.

Obtenidos los primos p y q se forma $N = p \times q$, y se halla la cantidad de números primos con N menores que N, que es dada por la función $\varphi(N) = (p-1) \times (q-1)$. φ es la función de Euler. $\varphi(N)$ cuenta la cantidad de números naturales menores que N que son primos con N. Si N fuera primo, $\varphi(N) = N - 1$. si $N = pq$, el producto de dos primos, su expresión es la antes descrita. Se elige e tal que $MCD(e, \varphi(N)) = 1$. El número e se elige al azar, aunque existen versiones que toman e fijo siendo un valor común $e = 2^{16} + 1$. El número $65537 = 2^{16} + 1$, tiene la particularidad de que en su representación binaria tienen sólo dos unos, siendo los otros dígitos iguales a cero, siendo - además- un número primo.

Como e es primo con $\varphi(N)$ existe su inverso d módulo $\varphi(N)$. En efecto, se cumple que existen d y h números enteros (pero no en general ambos positivos) tales que $d \cdot e + h \cdot \varphi(N) = MCD(e, \varphi(N)) = 1$. Estos números d y h pueden ser obtenidos por el AEE o el AEBE que se describen más adelante. Cada una de estas operaciones debe ser realizada con cuidados especiales para asegurar que el sistema sea seguro. Si, d resultara negativo, puede siempre tomarse $k \in I\!N$, $k > 0$, tal que $k \cdot \varphi(N) + d$ sea positivo, y este nuevo valor tomarlo como d. Una vez elegidos N, e, y d se hace pública la pareja de números (N, e) y se mantienen secretos a d, p, q, $\varphi(N)$. Para encriptar un mensaje M y enviárselo a $\mathbf{A}$, el usuario $\mathbf{B}$ busca en una guía *ad hoc* la clave pública de $\mathbf{A}$, (N_a, e_a) y le envía el mensaje $C = M^{e_a} \, mod \, N_a$. Al recibir C, el usuario $\mathbf{A}$, que conoce su clave secreta d_a, calcula $C^{d_a} \, mod \, N_a$ que es el mensaje M. En efecto, se cumple que si e y d son como antes $M = M^{e^d} \, mod \, N$. Pues lo anterior es equivalente, por el Teorema Chino del Resto (ver 1.8), a que simultáneamente se cumpla

$$\begin{aligned} M^{e^d} &\equiv M \mod p \\ M^{e^d} &\equiv M \mod q \end{aligned}$$

donde hemos escrito, como siempre, $N = p \cdot q$. Analicemos la primera ecuación, $M^{e^d} \equiv M \, mod \, p$, si M es múltiplo de p, entonces tanto M como M^{e^d} son 0 módulo p y la igualdad se cumple. Si no, por el Teorema de Fermat, se tiene que $M^{p-1} \equiv 1 \, mod \, p$. Pero entonces se tiene que, como $ed \equiv 1 \, mod \, \varphi(N)$, $ed = 1 + k\varphi(N) = 1 + k(p-1)(q-1)$. Resulta entonces que

$$M^{e^d} = M^{ed} = M^{1+k(p-1)(q-1)} = M \cdot (M^{p-1})^{k(q-1)} = M \, mod \, p$$

Análogamente se cumple $M^{e^d} \equiv M \, mod \, q$, de donde resulta que $M^{e^d} - M$ es múltiplo de p y de q. Como p y q son números primos distintos, son primos entre sí, por lo que resulta $M^{e^d} - M$ múltiplo del producto $pq = N$. Entonces M^{e^d} y M son el mismo número módulo N, como queríamos demostrar. La necesidad de mantener secretos a d, p, q, y $\varphi(N)$ se justifica en que:
Si se hace público el valor de d, entonces cualquiera puede desencriptar.
Si se hace público a p o a q, entonces se conoce a $\varphi(N) = (p-1)q - 1)$. Con esta información, aplicando el AEE, obtener d a partir de e, que es parte de la clave pública. Una vez en posesión de d podemos desencriptar cualquier mensaje.
Si poseemos $\varphi(N)$, calculamos d usando el AEE, y estamos como en el caso anterior.
En realidad, el conocer $\varphi(N)$ equivale a conocer p y q. Ya vimos que si conocemos p y q entonces $\varphi(N) = (p-1)(q-1)$. Pero si ahora conocemos $\varphi(N)$, entonces de la expresión $\varphi(N) = (p-1)(q-1) = pq - p - q + 1 = N - p - q + 1$ resulta $p + q = N + 1 - \varphi(N)$. Como también tenemos que $pq = N$, poseemos la suma y el producto de los números p y q. Esta información permite hallar los números p y q: ellos van a ser las dos raíces de la ecuación de segundo grado

$$x^2 - (N + 1 - \varphi(N))x + N = 0$$

Más aun, conociendo d y e podemos factorizar N.
En efecto: dados e y d calculamos $k = ed - 1$. Por definición de $\varphi(N)$ tenemos que $ed \equiv 1 \; mod \; \varphi(N)$ o sea que $k = ed - 1$ es múltiplo de $\varphi(N)$. Como $\varphi(N) = (p-1)(q-1)$ es par, entonces k es par y se puede escribir, $k = 2^t r$ con $t \geq 1$ y r impar. Tenemos que para todo $g \in \mathbb{Z}_N^*$ (g invertible en $\mathbb{Z}_N$) vale que $g^{\varphi(N)} \equiv 1$ (ver en 1.9.2 el Teorema de Euler) y por lo tanto $g^k \equiv 1$. Por lo tanto $g^{k/2}$ es una raíz de la unidad módulo N. Por el Teorema Chino del Resto 1 tiene cuatro raíces módulo $N = pq$. Dos de ellas

son 1 y $-1 = N - 1$. Las otras son x y $-x$, donde $x \equiv 1 \; mod \; p$ y $x \equiv -1 \; mod \; q$. Conociendo alguna de estas dos raíces, x o $-x$, podemos averiguar la factorización de N calculando $MCD(x - 1, N) = p$ (pues $x - 1$ va a ser múltiplo de p pero no de q). Se puede probar que eligiendo al azar $g \in \mathbb{Z}_N$ con probabilidad 1/2 (en la elección de g) uno de los elementos de la sucesión $g^{k/2}, g^{k/4}, \ldots, g^{k/2^t} \; mod \; N$ es una raíz de la unidad que coincide con x o $-x$. Todos los elementos de la sucesión pueden calcularse con el algoritmo de exponenciación rápida en forma eficiente.

Esto justifica la afirmación hecha más arriba, de que deben permanecer secretos los números d, p, q y $\varphi(N)$. Y más, indica que de ninguna manera puede darse a dos usuarios distintos el mismo valor de N (en principio algún administrador de red podría creer que conservando en secreto p y q y dando a varios usuarios solo los valores de e y d distintos pero manteniendo un N común 'en la red' existiría confidencialidad). En cambio sí puede darse el mismo valor de e a distintos usuarios.

1.5 Exponenciación rápida

El ejemplo más típico y más importante del principio anteriormente mencionado de la necesidad de controlar el crecimiento del tamaño de los números que intervienen en el cálculo con aritmética finita es la exponenciación módulo N. Se necesita un algoritmo eficiente para calcular a^k en $\mathbb{Z}_N$ donde a es un elemento de $\mathbb{Z}_N$, y k es un número entero, usualmente positivo. Se presentan un algoritmo, que llamaremos Pot1. Se describe en pseudo código.

1.5.1 Algoritmo Pot1

Deseamos calcular $a = b^n \, mod \, N$. Sea $n = (d_{k-1}d_{k-2}\cdots d_1 d_0)_2$, o sea $n = d_0 + 2d_1 + 4d_2 + \cdots + 2^{k-1}d_{k-1}$, cada d_j es 0 o 1.
Entrada: los números b, n, N. Salida: $a = b^n \, mod \, N$.

- 1 Hacer $a := 1$; $b := b \, mod \, N$; $j := 0$;
- 2 mientras $j < k = [\log_2 n] + 1$ hacer
- 3 Si $d_j = 1$ entonces hacer $a := a * b \, mod \, N$;
- 4 $b := b^2 \, mod \, N$;
- 5 $j := j + 1$; retornar a 2
- 6 fin.

Obsérvese que en cada paso se realizan a lo más 2 multiplicaciones con factores de a lo más $[\log_2 N] + 1$ de longitud. En total se realizan $k = [\log_2 n] + 1$ operaciones. Por lo tanto, el costo de este algoritmo es $O(\log_2 n \times \log_2^2 N)$.

1.6 Algoritmos para hallar el máximo común divisor

En varios problemas es necesario hallar el máximo común divisor entre números. Normalmente ello se implementa mediante el algoritmo de Euclides. Se da el pseudo-código correspondiente al cálculo de $D = MCD(a, b)$ juntamente con los valores de los enteros A y B tales que $Aa + Bb = D$. Este algoritmo se llama

1.6.1 Algoritmo de Euclides Extendido, AEE

Entrada: los números a, b que se suponen enteros positivos no nulos, $0 < b \leq a$.
Salida: el máximo común divisor D de a y b y números enteros A y B tales que $Aa + Bb = D$.

- 1 $a_1 := a$; $b_1 := b$; $x_0 := 1$; $y_0 := 0$; $x_1 := 0$; $y_1 := 1$;
- 2 mientras $b_1 \neq 0$ hacer
- 3 $q := a_1 \, div \, b_1$; $x_{aux} := x_1$; $y_{aux} := y_1$; $x_1 := x_0 - q * x_1$; $y_1 := y_0 - q * y_1$; $x_0 := x_{aux}$; $y_0 := y_{aux}$;
- 4 $b_{aux} := b_1$; $b_1 := a_1 \, mod \, b_1$; $a_1 := b_{aux}$;
- 5 retornar a 2
- 6 $D := a_1$; $A := x_1$; $B := y_1$
- 7 fin.

Es importante notar que los números decaen a la mitad (al menos) a cada dos pasos. En efecto: si el resto $r_{i+1} \leq \frac{1}{2} r_i$ como $r_{i+2} < r_{i+1}$ entonces $r_{i+2} < \frac{1}{2} r_i$. Pero si $r_{i+1} > \frac{1}{2} r_i$, como $r_{i+1} < r_i$, entonces al dividir r_i por $r_{i+1} \neq 0$ se tiene cociente igual a 1 y resto $r_{i+2} = r_i - r_{i+1} < r_i - \frac{1}{2} r_i = \frac{1}{2} r_i$. Así que siempre se obtiene $r_{i+2} < \frac{1}{2} r_i$. El número de operaciones a realizar es entonces de la magnitud de $log_2 a$, pues por cada paso se realizan
1 división 2 multiplicaciones 1 resto (módulo) y 2 sustracciones. La complejidad de cada paso es entonces a lo más $O((\log_2 a)^2)$ ya que cada término en las operaciones es menor que a, luego su longitud es menor que la de a. Pero el número de iteraciones del algoritmo, porque $r_{i+2} < \frac{1}{2} r_i$, está acotado por $2 \log_2 a$.

Por lo tanto la complejidad del AEE es $O((\log_2 a)^3)$. Al retener los resultados intermedios, el algoritmo sirve para hallar los coeficientes de la combinación lineal entera que da el MCD, esto es A y B tales que $A \times a + B \times b = D = MCD(a, b)$. En caso de que a y b sean primos entre sí sirve para hallar el inverso de a módulo b, o viceversa, es decir para hallar el inverso en cierta aritmética finita. En efecto, si $D = 1$, entonces $Aa + Bb = 1$, o sea que $Aa \equiv 1 \, mod \, b$.

1.6.2 Algoritmo Binario Extendido, ABE

Un método alternativo de hallar el $MCD(a, b)$ se basa en la división por 2, que es fácilmente implementable en un ordenador y equivale a hacer un shift de los datos a la derecha. Se observa que:

- $MCD(2a, 2b) = 2 \times MCD(a, b)$
- si a es impar $MCD(a, b) = MCD(a, 2b)$.
- $MCD(a, b) = MCD(b, a - b)$
- $MCD(a, a) = a$ y $MCD(a, 0) = a$ si $a \neq 0$

El pseudocódigo siguiente corresponde al Algoritmo de Euclides Binario Extendido. Da, conociendo $a \geq b > 0$ enteros, los valores de $D = MCD(a, b), A, B$, enteros, tales que $Aa + Bb = D$. **Asumimos ahora que tanto a como b son impares**.

1. Hacer ($r_0 := a$; $r_1 := b$; $x_0 := 1$; $y_0 := 0$; $x_1 := 0$; $y_1 := 1$; $expo := 0$)
2. si $r_0 <> r_1$ hacer ($r_2 = r_0 - r_1$; $x_2 := x_0 - x_1$; $y_2 := y_0 - y_1$;)
3. Mientras r_2 sea par hacer ($r_2 := r_2/2$; $x_2 := x_2/2$; $y_2 := y_2/2$; $expo := expo + 1$;)
4. Si $r_2 > r_1$ mover ($r_2 \rightarrow r_0$; $x_2 \rightarrow x_0$; $y_2 \rightarrow y_0$;)
5. Si $r_2 \leq r_1$ mover ($r_1 \rightarrow r_0$; $x_1 \rightarrow x_0$; $y_1 \rightarrow y_0$; $r_2 \rightarrow r_1$; $x_2 \rightarrow x_1$; $y_2 \rightarrow y_1$;)
6. Si (($r_0 = r_1$) o ($r_1 = 1$)
   entonces retornar(r_1,$expo$,x_1,y_1);
   sino volver al paso 2.

A la salida tenemos expresado $r_1 = D = MCD(a,b) = x_1 a + y_1 b$, solo que x_1 e y_1 no son, en general, enteros. Si $expo = k$ tenemos que $x_1 = \frac{h}{2^k}$ y también $y_1 = \frac{l}{2^k}$. Estos denominadores 2^k aparecen en el paso 3 por cada vez que se divide a x_2 y a y_2 por 2.
Pero nos interesa escribir D como $D = Aa + Bb$ con A y B enteros. Observemos que para todo u entero vale

$$D = x_1 a + y_1 b = \frac{h + ub}{2^k} a + \frac{l - ua}{2^k} b$$

Entonces si conseguimos hallar u que haga que a la vez $h+ub$ y $l-ua$ sean múltiplos de 2^k tendremos solucionado el problema.
Queremos entonces resolver $h + ub \equiv 0 \ mod \ 2^k$ y $l - ua \equiv 0 \ mod \ 2^k$. De la primera ecuación resulta $u = -hb^{-1} \ mod \ 2^k$. El problema entonces se reduce a calcular $b^{-1} \ mod \ 2^k$. Supongamos que esto esta logrado. Entonces

$$D - \frac{h + ub}{2^k} a = \frac{l - ua}{2^k} b$$

Como el termino de la izquierda en la ultima igualdad es entero resulta que también lo es el de la derecha. Por lo tanto también se cumple para el u hallado que $l - ua \equiv 0 \ mod \ 2^k$. Queda entonces solamente dar un algoritmo rápido para calcular $b^{-1} \ mod \ 2^k$.

Algoritmo para calcular b^{-1} mod 2^k.
entrada: k y b impar que puede suponerse menor que 2^k
salida: $y = b^{-1}$ inverso de b mod 2^k.

1. hacer $y := 1$;
2. para $i := 2$ hasta k hacer (calcular $r = by \ mod \ 2^i$; si $2^{i-1} < r$ entonces $y := y + 2^{i-1}$);
3. retornar(y)

Justificación del algoritmo anterior.
Observemos que si $k \leq 3$ entonces $b^{-1} \ mod \ 2^k$ es $b \ mod \ 2^k$. Si $k = 1$ es claro que $b \equiv 1 \equiv b^{-1} \ mod \ 2$. Si $k = 2$ entonces $b \equiv 1$ módulo 4 o $b \equiv 3$. En cualquier caso $b^2 \equiv 1$ módulo 4. Del mismo modo si $k = 3$ y b es impar resulta $b \equiv 1$ o $b \equiv 3$ o

$b \equiv 7$ módulo 8 y resulta $b^2 \equiv 1$ módulo 8, o sea $b = b^{-1}$ módulo 8.
Esto dice que siempre podríamos elegir $y = b^{-1} = b$ si $1 \leq k \leq 3$ y empezar el algoritmo a partir de $i := 4$ hasta k. En realidad optamos por definir solo el caso $k = 1$ haciendo $y = 1$. Si en un paso i genérico se tiene que ya y cumple que $by \equiv 1 \ mod \ 2^{i-1}$ entonces $by = 1 + h2^{i-1}$. Puede pasar que $by \equiv 1 \ mod \ 2^i$ o que $by \equiv 1 + 2^{i-1} \ mod \ 2^i$ según sea h par o impar. Si pasa lo primero nada hay que hacer y dejamos el mismo valor de y que en el paso $i - 1$. Si pasa lo segundo hay que corregir el resultado sumándole al valor de y el entero 2^{i-1}. Si $b = b_{k-1}2^{k-1} + b_{k-2}2^{k-2} + \cdots + b_1 2 + 1$ queda para algún entero s:

$$b(y+2^{i-1}) = by+b2^{i-1} = 1+h2^{i-1}+(b_{k-1}2^{k-1}+b_{k-2}2^{k-2}+\cdots+b_1 2)2^{i-1}+1\times 2^{i-1} =$$

$$1 + (h+1)2^{i-1} + s2^i = 1 + (\frac{h+1}{2} + s)2^i \equiv 1 \ mod \ 2^i$$

El paso 3 del algoritmo justamente hace eso, corrige el valor de y sumándole 2^{i-1} en caso de que $by \equiv 1 + 2^{i-1} \ mod \ 2^i$ ya que en ese caso $2^{i-1} < r = 1 + 2^{i-1}$.

1.7 Aritmética de Montgomery

Al utilizar en Criptografía a los enteros módulo p, con p primo, normalmente hay que sumar $x + y$, restar $x - y$, multiplicar xy, multiplicar por el inverso xy^{-1} y exponenciar x^n, con n entero. Todas las operaciones módulo p. Los resultados se escriben usualmente como números entre 0 y $p - 1$. Para ello en la aritmética usual se realizan las operaciones usuales en $\mathbb{Z}$ y se reducen a $\mathbb{Z}_p$ dividiendo entre p y calculando el resto. La gran cantidad de divisiones es sin embargo una de las causas del enlentecimiento de los cálculos. Para el caso de sumas o restas esto no causa grandes problemas, basta con sumar o restar p a $z = x \pm y$ para obtener una representación de z entre 0 y $p - 1$. La situación cambia si consideramos xy o xy^{-1}. En estos casos los números manejados pueden ser del orden de p^2 y el calcular $xy \ mod \ p$ involucra una división con un orden de $\log_2 p$ operaciones de bit. Al aplicar el algoritmo de exponenciación rápida módulo p, hay que realizar del orden de $\log_2 n$ divisiones por p lo que da unas $\log_2 p \log_2 n$ operaciones de bit. Si p tiene unos 1000 bits y aplicamos un test de primalidad como el de Fermat, deberemos elevar a a la potencia $p - 1$ módulo p. Esto da unas 10^6 operaciones de bit, sin contar las multiplicaciones involucradas.

La representación de Montgomery busca subsanar este problema. Sea p un primo impar dado y b la base en que representamos los números. En un ordenador podemos asumir que $b = 2$ o $b = 2^k$. Sean t y R definidos por $R = b^t > p > b^{t-1}$. Es claro que $MCD(R, p) = 1$.

Definición 1.7.1 *La representación de Montgomery de $x \in \mathbb{Z}_p$ se obtiene calculando $y \equiv xR \ mod \ p$.*

Es claro que esta representación es unívoca: dado $y \in \mathbb{Z}_p$ existe un único $x \in \mathbb{Z}_p$ tal que $y \equiv xR \ mod \ p$ lo que es equivalente a $x \equiv yR^{-1} \ mod \ p$.
La suma y resta en esta representación se realiza como usualmente: $xR + zR = (x + y)R$. Si además $y \equiv xR \ mod \ p \ w \equiv zR \ mod \ p$ cumplen $0 \leq y \leq p - 1$, $0 \leq w \leq p - 1$ la suma $xR + zR \ mod \ p$ es a lo sumo es del orden de $2(p - 2)$ y restando p conseguimos un representante entre 0 y $p - 1$ (observar que $p = pR = 0$ módulo p). No es preciso realizar divisiones.
Al realizar la multiplicación de xR por zR obtenemos xzR^2. Querríamos multiplicar por R^{-1} módulo p para obtener la representación de Montgomery.
Sea y tal que $0 \leq y < pR$.

Lema 1.7.1 *Definamos $u = -yp^{-1} \ mod \ R$. Sea $x = \frac{y+up}{R}$. Entonces x es un entero, $0 \leq x < 2p$ y $x = yR^{-1} \ mod \ p$.*

Demostración:El número $y + up = y(1 - p^{-1}p) \equiv 0 \ mod \ R$ es claro que es múltiplo de R. Así que $x = \frac{y+up}{R} \in \mathbb{Z}$. Además $xR = y + up \equiv y \ mod \ p$. O sea, $x \equiv yR^{-1} \ mod \ p$. Por último, como $0 \leq y < pR$ y $0 \leq u < R$ se tiene: $y + up < pR + Rp = 2pR$ Dividiendo por R se tiene que $x < 2p$.

■

Esto sugiere que para multiplicar xR por zR en $\mathbb{Z}_p$ y escribirlo en la representación de Montgomery hagamos primero la multiplicación en $\mathbb{Z}$. Luego, como las representaciones $[xR], [zR]$ de xR y de zR en $\mathbb{Z}_p$ cumplen $0 \leq [xR] < p$, $0 \leq [zR] < p$, resulta $0 \leq [xR][yR] < p^2 < pR$ y podemos aplicar el lema anterior con $y = [xR][zR]$ para obtener $yR^{-1} = xzR \ mod \ p$. Así no es preciso dividir para obtener la representación de Montgomery, pues el lema dice que xzR cumple $0 \leq xzR < 2p$. Si $xzR \geq p$ restando p obtenemos una representación de xzR en

$\mathbb{Z}_p$ entre 0 y $p-1$.
A seguir damos los algoritmos correspondientes.
Primero es bueno observar que si $R = 2^k$ el cálculo de $p^{-1}\ mod\ R$ puede hacerse utilizando el algoritmo dado en 1.6.2 sin necesidad de divisiones. Si ese no fuera el caso, de todos modos $p^{-1}\ mod\ R$ debe calcularse solo una vez. Es bueno observar también que representando los números en la base b, calcular $z\ mod\ b^t$ se realiza con un *shift* a la izquierda de t lugares en la representación de z en la base b. No es, por lo tanto, necesario realizar divisiones.

Algoritmo de reducción de Montgomery

Suponemos dados p primo y $R = b^t > p > b^{t-1}$ donde b es la base de numeración usada.
Entrada: un número y entre 0 y pR.
Salida: $x = yR^{-1}\ mod\ p$

1. hallar $u := -yp^{-1}\ mod\ R$.
2. Calcular $x := \frac{y+up}{R}$.
3. Si $x \geq p$ hacer $x := x - p$.
4. Retornar(x).

La división por $R = b^t$ del cálculo de x es también realizable con un *shift* a izquierda de t lugares.
Para multiplicar x por z en $\mathbb{Z}_p$ usando la representación de Montgomery, xR, zR, debemos hallar xzR. Para ello hacemos

1. Multiplicamos xR por zR en $\mathbb{Z}$.
2. Aplicamos el Algoritmo de Reducción de Montgomery a xzR^2 obteniendo $xzR\ mod\ p$.

Si queremos reobtener la representación estándar en $\mathbb{Z}_p$ aplicamos a $xzR\ mod\ p$ el Algoritmo de Reducción de Montgomery obteniendo $xz\ mod\ p$.
Para obtener la representación de Montgomery como un numero entre 0 y $p-1$ a partir de $x \in \mathbb{Z}_p$, debemos calcular $xR\ mod\ p$ a partir de x. Para eso usamos los mismos algoritmos computando primero $R^2\ mod\ p$ lo que se hace una sola vez.

Observar que si $R = 2^t > p > 2^{t-1}$ entonces $R < 2p$ y la representación de $R \; mod \; p$ entre 0 y $p-1$ es $R-p$. Calculada $U = R^2 \; mod \; p$ para hallar $xR \; mod \; p$ calculamos xU en $\mathbb{Z}$ y luego le aplicamos el Algoritmo de reducción de Montgomery que nos da $xR \; mod \; p$.

Observación 1.7.1 *Observar que en muchos casos no se necesita calcular xR, dado que se parte de elegir al azar x entre 0 y $p-1$ y da lo mismo hacerlo directamente con xR.*

Para invertir $X = xR \; mod \; p$ en la representación de Montgomery aplicamos el Algoritmo de Multiplicación de Montgomery a xR dos veces obteniendo xR^{-1}. Luego con el AEE o, mejor aun, con el ABE ya que **no** queremos hacer divisiones, obtenemos $(xR^{-1})^{-1} \equiv x^{-1}(R^{-1})^{-1} \equiv x^{-1}R \; mod \; p$.

1.7.1 Función φ de Euler

Dado un número N la función de Euler es el valor de

$$\varphi(N) = \#\{x \,/\, 1 \leq x \leq N \, MCD(x, N) = 1\}$$

O sea, la cantidad de números entre 1 y N primos con N. Se demuestra que si la descomposición en factores primos de N es $N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, entonces $\varphi(N) = N(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k})$. Por ejemplo, si $N = 36 = 2^2 \times 3^2$ entonces $\varphi(36) = 36(1 - \frac{1}{2})(1 - \frac{1}{3}) = \frac{36 \times 2}{2 \times 3} = 12$.
Esto significa que entre 1 y 36 hay 12 números primos con 36. Estos son los números 1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35.
No confundir *número primo*, con *número primo con otro*. Así, $35 = 5 \times 7$ es primo con 36 pues $MCD(35, 36) = 1$, aunque no es primo al ser el producto de 5 por 7.

1.8 Teorema Chino de los Restos

Supongamos que queremos resolver el sistema de ecuaciones

$$\begin{array}{llll} x & \equiv & a_1 & mod \quad m_1 \\ x & \equiv & a_2 & mod \quad m_2 \\ \cdots & \cdots & \cdots & \qquad \cdots \\ x & \equiv & a_k & mod \quad m_k \end{array}$$

y que se cumple que cada pareja m_i, m_j son primos entre sí, o sea que $MCD(m_i, m_j) = 1$ si $i \neq j$. Entonces existe una única solución de todas ellas módulo M donde $M = m_1 m_2 \cdots m_k$. Vale decir que existe un único número entero H entre 0 y $M-1$ que las resuelve simultáneamente a todas. Este es el contenido del Teorema Chino de los Restos. Para ver que la solución debe ser única, supongamos que hay dos, x_p y x_s. Entonces $x = x_p - x_s$ es solución del sistema de congruencias

$$\begin{array}{lllll} x & \equiv & 0 & mod & m_1 \\ x & \equiv & 0 & mod & m_2 \\ \cdots & \cdots & \cdots & & \cdots \\ x & \equiv & 0 & mod & m_k \end{array}$$

Esto equivale a que x debe ser múltiplo de m_i para todo i. Como todos los m_i son primos entre sí resulta que x es múltiplo de M. Pero entre 0 y $M-1$ sólo el 0 es múltiplo de M. Luego $x = 0$ y entonces $x_p = x_s$. Para hallar una solución definimos números $M_i = M/m_i$. Entonces M_i es múltiplo de m_j para todo $j \neq i$, y $MCD(M_i, m_i) = 1$. Por el algoritmo AEE hallamos N_i y n_i tales que $MCD(M_i, m_i) = 1 = M_i N_i + m_i n_i$ o sea que N_i es el inverso de M_i módulo m_i. Hagamos

$$H = \sum_i a_i M_i N_i = a_1 M_1 N_1 + a_2 M_2 N_2 + \cdots + a_k M_k N_k$$

Tomando este número módulo M, obtenemos la solución buscada. En efecto, al tomar $H \, mod \, m_j$ se tiene que en la suma se anulan todos los sumandos excepto el $a_j M_j N_j$. Esto es así pues para $i \neq j$ M_i es múltiplo de m_j, luego $a_i M_i N_i$ también, así que son congruentes con 0 módulo m_j. Pero para $i = j$ $M_j N_j \equiv 1 \, mod \, m_j$. Entonces $a_j M_j N_j \equiv a_j \, mod \, m_j$. O sea, $H \equiv a_j \, mod \, m_j$.

■

En el caso de que tengamos sólo dos números p y q primos entre sí, como en el problema planteado por el algoritmo RSA que se ve en la sección 1.4.2, queda el sistema,

$$x \equiv a \, mod \, p$$

$$x \equiv b \, mod \, q$$

La solución en este caso se calcula como

$$H = apr + bps \, mod \, pq$$

donde r y s son enteros hallados por el AEE, tales que $qr \equiv 1 \, mod \, p$ y $ps \equiv 1 \, mod \, q$.

El teorema chino de los restos aparece por primera vez mencionado en el *Manual de aritmética del maestro Sun* escrito entre el 300 y el 400 d.C., vinculado con aplicaciones a la Astronomía.

Damos ahora un método para partir una clave entre varias personas. Por ejemplo: para poder poner en marcha una ojiva nuclear, se necesita la autorización del presidente de Estados Unidos y de al menos dos otros jefes del Pentágono. Esto es a los efectos de impedir que un lunático lance un ataque nuclear por su cuenta. El compartir la clave entre varios torna esto más difícil.

Supongamos que deseamos que de un grupo de n personas, k de ellas sean capaces de reconstruir una clave cuando están juntas pero menos de k no lo sean. Para ello elegimos n enteros positivos $0 < m_1 < m_2 < \cdots < m_n$ tales que son primos entre sí, o sea $MCD(m_i, m_j) = 1$ si $i \neq j$. A continuación definimos N como el producto de los k menores números m_j, $N = m_1 m_2 \cdots m_k$, y K como el producto de los $k-1$ mayores de entre los m_j. Exigimos además que N sea mayor que K y elegimos s entre ambos. O sea $K < s < N$. Resolvemos las congruencias

$$a_1 \equiv s \, mod \, m_1, \ldots, a_n \equiv s \, mod \, m_n$$

A cada uno de los n individuos se les asigna una pareja (a_j, m_j), $j = 1, \ldots, n$. Si ahora están presentes k personas o más, el número M que pueden formar multiplicando los números m_j que poseen cada uno de ellos es mayor que s. Pero con menos de k personas eso no es posible, dada la elección de K y N. Para reobtener el número s, deben resolver el sistema de congruencias

$$\begin{array}{cccc} x & \equiv & a_{h_1} \quad mod & m_{h_1} \\ x & \equiv & a_{h_2} \quad mod & m_{h_2} \\ \cdots & \cdots & \cdots & \cdots \\ x & \equiv & a_{h_k} \quad mod & m_{h_k} \end{array}$$

donde h_i es alguno de los índices de 1 a n. Por el Teorema Chino de los Restos existe solución única módulo M, pero s es menor que M y es solución, luego es *la* solución.

Veamos un ejemplo sencillo: Tomamos los números 11, 13, 17, 21, 25, 31, $N = 11 \times 13 \times 17 = 2431$ y $K = 25 \times 31 = 775$. Escogemos como llave secreta s a un número entre 775 y 2431 por ejemplo $s = 983$. Entonces se tiene que $983 \equiv 4 \, mod \, 11$, $983 \equiv 8 \, mod \, 13$, $983 \equiv 14 \, mod \, 17$, $983 \equiv 17 \, mod \, 21$, $983 \equiv 8 \, mod \, 25$, $983 \equiv 22 \, mod \, 31$. Así las claves a repartir serían

$$(11, 4), \, (13, 8), \, (17, 14), \, (21, 17), \, (25, 8), \, (31, 22)$$

Por supuesto que el conocimiento de una sola clave hace muy difícil adivinar s, y aún con dos claves el valor exacto no puede saberse. Pero tres personas recuperan el valor aplicando el Teorema Chino de los Restos.

1.9 Tests de primalidad

Sistemas como el RSA se basan en que es muy difícil descomponer en factores primos a un número compuesto si éste es muy grande, digamos, el producto de dos primos enormes. Pero, ¿cómo saber si un número es primo sin descomponerlo? Parecería que estamos frente a una dificultad equivalente a la que queremos poner a los posibles transgresores de la privacidad. Pues si el número p es del orden de las 200 cifras decimales, factorizarlo, con los métodos más veloces conocidos y las computadoras más poderosas es una tarea que lleva un tiempo enorme, si este fuera el criterio para saber si es primo un numero tardaríamos meses en armar una sola clave. Si ascendemos a 800 cifras decimales nos encontramos que esa tarea no puede realizarse aunque para ello dispusiéramos del tiempo de vida del Sol.

¿Cómo superar entonces el escollo? Afortunadamente saber si un número es primo es más fácil que descomponer un número en factores primos.

Para chequear si un número es primo se dispone de herramientas que no implican factorizar el número. La más sencilla es un clásico teorema de Fermat.

Teorema 1.9.1 *Teorema de Fermat*
Si p es un número primo entonces para todo a, $1 \leq a < p$ vale que el resto de dividir a^{p-1} por p es 1. En aritmética módulo p eso es que $a^{p-1} \equiv 1 \, mod \, p$.

Demostración:
Como p es primo, entonces $\mathbb{Z}_p$ es un cuerpo (cada elemento distinto de 0 es invertible puesto que en $\mathbb{Z}_N$, con $N \in \mathbb{N}$, $N > 0$ que a sea invertible equivale a

$MCD(a, N) = 1$ y si p es primo entonces todo número entre 1 y $p-1$ cumple esa condición). Para J desde 1 hasta $p-1$ hacemos variar J definiendo la función $f(J) = a \cdot J \, mod \, p$. Esta función es inyectiva puesto que si $f(J) = f(I)$ entonces $a \cdot J = a \cdot I \, mod \, p$ de aquí resulta que $a \cdot (J - I) = 0 \, mod \, p$. Como $a \neq 0 \, mod \, p$ es invertible, o sea que existe su inverso b tal que $b \cdot a = 1 \, mod \, p$ (Este valor b puede calcularse con el AEE, por ejemplo). Multiplicando por b a ambos lados de $a \cdot (J - I) = 0 \, mod \, p$ obtenemos que $b \cdot a \cdot (J - I) = b \cdot 0 \, mod \, p$ o sea $1 \cdot (J - I) = J - I = 0 \, mod \, p$. Se concluye que $J = I$ por lo que f es inyectiva. Pero $\mathbb{Z}_p$ es un conjunto finito y una función inyectiva en un conjunto finito también es sobreyectiva. Por lo tanto resulta que al J recorrer todos los valores posibles $J = 1, \cdots, p-1$ de $\mathbb{Z}_p$ también $f(J)$ hace lo mismo solo que posiblemente en un orden distinto. Multiplicando todos los valores de J entre si tenemos

$$1 \times 2 \times 3 \times \cdots \times p - 1 \, mod \, p \qquad (*)$$

que será igual módulo p, por lo que dijimos recién, al producto

$$f(1) \times f(2) \times f(3) \times \cdots \times f(p-1) \, mod \, p =$$

$$a1 \times a2 \times a3 \times \cdots \times a(p-1) \, mod \, p =$$

$$a^{p-1} \times 1 \times 2 \times 3 \times \cdots \times p - 1 \, mod \, p \qquad (X)$$

Igualando $(*)$ con (X) y simplificando obtenemos

$$a^{p-1} = 1 \, mod \, p$$

Esto es lo que queríamos demostrar.

■

Teorema 1.9.2 *Teorema de Euler*
Si n es un número entero positivo, entonces para todo a, $1 \leq a < n$, tal que $MCD(a, n) = 1$, vale que el resto de dividir $a^{\varphi(n)}$ por n es 1. En aritmética módulo n eso es que $a^{\varphi(n)} \equiv 1 \, mod \, n$.

Demostración:
La demostración es mu parecida a la ya vista del Teorema de Fermat. Solo hay que definir la función $f(J) = a \cdot J \, mod \, n$ para J variando en los invertibles de $\mathbb{Z}_n$ en lugar de en todos los numeros entre 1 y $n-1$.

■

El teorema de Fermat permite descartar candidatos a primos que no lo son. Pues si n es el número que queremos saber si es primo y resulta que $a^{n-1} \not\equiv 1 \, mod \, n$, entonces n no es primo.
Lo ideal sería que valiera que si n es compuesto este test siempre lo detectara. Desafortunadamente no es así, y existen números, conocidos por números de Carmichael que se portan como primos sin serlo. Se los llama pseudoprimos.

1.9.1 Números de Carmichael

Los números de Carmichael son aquellos enteros N que cumplen con el pequeño Teorema de Fermat, o sea que

$$a^{N-1} \equiv 1 \, mod \, N$$

aunque no son primos. Se puede probar (Teorema de Korselt) que un entero positivo es un número de Carmichael si y sólo si cada factor primo p de N cumple

- p^2 no divide N. (O sea que en la descomposición factorial de N todos los primos que aparecen lo hacen a la primera potencia.)
- $p-1$ divide $N-1$.

Se puede probar que en un número de Carmichael aparecen por lo menos 3 factores primos distintos. El número de Carmichael más pequeño es $3 \times 11 \times 17 = 561$. Aunque son números bastante raros, en el sentido de que su proporción es pequeña en relación al conjunto de los números naturales, se probó en 1994 que existen infinitos números de Carmichael. De todos modos el test de Fermat se lo utiliza para detectar si un numero es primo o no. Existen otros tests más finos que también se usan y para los que no hay números de Carmichael, vale decir que si uno pudiera probar esos tests en **todos** los casos posibles, y el test no fallara,

entonces tendríamos certeza de que es un primo. Si un número n no pasa el test de Fermat u otro cualquiera sabemos que es compuesto. Si pasa el test, **no** sabemos si es primo, sabemos que es primo con una cierta probabilidad. Tomando distintos valores de a entre 1 y $n-1$ se puede probar que podemos lograr que esa probabilidad sea tan alta como se quiera aumentando el número de veces que se aplica el test. RSA Data Security declara (1997) que primero prueba si el número es divisible por primos chicos, digamos los primos menores que 10000. y luego aplica el test de Fermat unas 4 veces para números a elegidos al azar. Si luego de esto no se detectó que el número fuese compuesto, la probabilidad de que lo sea es menor a 10^{-10}, lo que se considera despreciable.
En la práctica consideramos que un tal número es primo. ¿ Y si no lo es? En ese caso lo más probable es que al descifrar un mensaje se obtengan resultados incorrectos y se decida cambiar la clave.

1.10 Test de primalidad de Miller-Rabin

Supongamos que n es un natural muy grande del que deseamos saber si es primo o no. Supongamos además que al elevar b a la $n-1$ nos da congruente con 1 módulo n (ie: n es pseudoprimo respecto a la base b). La idea detrás del criterio de primalidad de Miller-Rabin es que si sucesivamente sacamos "raíces cuadradas[a] la congruencia $b^{n-1} \equiv 1 \ mod \ n$, o sea, si elevamos a b a las potencias $\frac{n-1}{2}, \frac{n-1}{4}, \ldots, \frac{n-1}{2^s}$ modulo n, donde $t = \frac{n-1}{2^s}$ es impar, entonces el primer residuo módulo n distinto de 1 *debe* ser -1. Esto porque si n es primo las únicas raíces cuadradas de 1 son 1 y -1. En la práctica operamos al revés: escribimos $n = 2^s t$ con t impar. Luego elegimos b al azar entre 1 y $n-1$ y calculamos $b^t \ mod \ n$. Si es $b^t \equiv 1 \ mod \ n$ paramos, si no elevamos a b^t al cuadrado calculando $(b^t)^2 = b^{2t} \ mod \ n$, y así continuamos elevando al cuadrado y calculando $(b^{2t})^2 = b^{2^2 t} \ mod \ n$, etc. hasta obtener $b^{2^{j-1}t} \equiv -1 \ mod \ n$ y por lo tanto $b^{2^j t} \equiv 1 \ mod \ n$, con $j \leq s$, o llegar a $j = s-1$ sin obtener $b^{2^{s-1}t} \equiv -1 \ mod \ n$. En este caso sabemos que n es compuesto. Se puede probar que si n es compuesto **por lo menos** para el 75% de los b entre 1 y $n-1$ el test no pasa. Eligiendo al azar b tenemos un $\frac{1}{4}$ de casos a lo más en que el test pasa siendo n compuesto. Si elegimos $b_1, b_2, \ldots b_k$ en forma independiente entre 1 y $n-1$ la probabilidad de que el test **no** detecte que n es compuesto es menor o igual que $\frac{1}{4^k}$.

A continuación damos en pseudocódigo la descripción del Test de Rabin-Miller.

- Procedimiento RABIN(entrada: n: entero impar); salida: **falso** si n es compuesto, **verdadero** si n es un número primo (*con probabilidad* $1 - \frac{1}{4}$).
- Paso 1: leer n;
- Paso 2: hallar s tal que $n - 1 = 2^s t$ con t impar; hacer $i := 0$;
- Paso 3: elegir b al azar en el intervalo $[2, n-2]$
- Paso 4: hacer $b := b^t \, mod \, n$; si $b = 1$ módulo n return(**verdadero**)
- Paso 5: si $b = n - 1$ módulo n return(**verdadero**)
- Paso 6: hacer $i := i + 1$; si $i < s$ hacer $b := b^2$ e ir a paso 5; si no return(**falso**)

Como dijimos, la repetición del test k veces da un margen de error de $\frac{1}{4^k}$ de que un número sea declarado primo siendo que no lo es.

1.11 Generación de números primos

La generación de números primos es de vital importancia en sistemas de clave pública como el RSA o el de Rabin o el de logaritmo discreto sobre $\mathbb{Z}_p$. Diversas precauciones deben tomarse para conseguir seguridad en los sistemas contra el criptoanálisis. Gran parte de la seguridad, tanto en el sistema RSA como en el de Rabin y los basados en logaritmo discreto y otras variantes análogas, reside en la dificultad de factorizar números grandes.

O sea, dado N el producto de dos números primos p y q, $N = pq$, no podemos hallar ni p ni q a partir de N, en un tiempo razonable, ni aún contando con los recursos más poderosos en Computación ni los métodos más avanzados en Teoría de Números. Pero para lograr este propósito no alcanza con elegir primos grandes, éstos deben a su vez cumplir requisitos adicionales. Entre los más importantes se destacan:

- p y q deben ser grandes, del orden de al menos 512 bits cada uno. Si se desea tener seguridad por muchos años el número de dígitos debe duplicarse. Deben ser ambos de aproximadamente del mismo número de bits.

- La diferencia entre p y q debe ser grande, digamos del orden de los 400 bits.

- La descomposición factorial de $p-1$, $p+1$, $q-1$ y $q+1$, debe contener entre sus factores un número primo grande. Por ejemplo, números primos de Mersenne, de la forma $p = 2^k - 1$ no son aceptables, pues $p+1 = 2^k$. Un ejemplo de un tal número primo es $p = 2^{756839} - 1$. Cuando pensamos que $p-1$ y $p+1$ tengan en su descomposición factorial un primo p' grande, estamos diciendo que p' debe ser del orden de los 150 bits.

- El mínimo común múltiplo entre $p-1$ y $q-1$ debe ser grande. Esto puede lograrse si el primo p' grande, que aparece en la descomposición de $p-1$ es distinto del primo q' grande que aparece en la de $q-1$.

- En el sistema RSA, el menor número positivo que hace que al iterar el proceso de encriptado P con la clave e de encriptar, se vuelva a repetir el mensaje, debe ser grande (como estamos trabajando con un número finito de símbolos, *siempre* existe un k tal que c coincide con $P^k(c)$, donde escribimos $P^k(c)$ para representar la iteración $P(P(\cdots P(c)\cdots))$ donde el proceso de encriptado P se aplica k veces). Este número k va a ser grande si, por ejemplo, se pide que siendo p'' y q' los factores primos grandes de $p-1$ y $q-1$ antes mencionados, a su vez $p'-1$ y $q'-1$ contienen factores primos p'' y q'' respectivamente, tales que siendo e la clave de encriptar, entonces

$$e^{(p'-1)/p''} \neq 1 \, mod \, p', \qquad e^{(q'-1)/q''} \neq 1 \, mod \, q'$$

  con $p'' \neq q''$ y siendo $p''q''$ grande, digamos de 100 bits.

En este momento, el lector puede preguntarse si existen números p y q primos que cumplan con todas estas restricciones. Y en caso de existir cuán abundantes son, y cuán difícil es su búsqueda. Por eso la siguiente información tal vez no carezca de interés.

- Existen infinitos números primos.

- La probabilidad de que eligiendo al azar un número entre 1 y n, para n grande, éste sea primo es aproximadamente de $\frac{1}{\log n}$. O sea que el número aproximado de primos entre 1 y n es de $\frac{n}{\log n}$. Por lo tanto entre 10^{90} y 10^{110}, suponiendo

que elegimos tener primos de aproximadamente 100 dígitos, existen más de 10^{100} primos. En efecto,

$$\frac{10^{110}}{\log(10^{110})} - 10^{90} > 10^{105} - 10^{90} = (10^{15} - 1) \times 10^{90} > 10^{100}$$

Se calcula que existen 10^{77} átomos en el universo conocido, así que asignándole a cada átomo un número primo por cada segundo desde el nacimiento del universo, hace unos 10^{12} años, aún sobrarían.

- Dada una progresión aritmética de la forma $kn + h$, $n = 0, 1, 2, \ldots$ con k y h enteros fijos, si $MCD(k, h) = d > 1$ es claro que no puede haber más de un número primo en la progresión. Dirichlet probó que en caso de que $MCD(h, k) = 1$ existen infinitos números primos en la progresión aritmética $kn + h$. Dicho de otro modo, existen infinitos primos p congruentes con h módulo k si $MCD(k, h) = 1$.

- Más aún: si n es grande, el número de primos congruentes con $h \, mod \, k$ es aproximadamente $\frac{1}{\varphi(k)} \frac{n}{\log n}$, donde $\varphi(k)$ es la función de Euler de k. ¿Cuántos números primos congruentes con 2 módulo 3 hay entonces entre 10^{90} y 10^{110}? Aproximadamente la mitad de todos, pues $\varphi(3) = 2$. Eso dice que hay más de 10^{100} también en este caso. Análogas consideraciones pueden hacerse cuando se buscan primos de la forma $4n + 3$ o de la forma $8n + 3$ u $8n + 7$. En el último caso, por ejemplo, como $\varphi(8) = 4$, resulta que puede esperarse que 1 de cada 4 primos encontrados al azar sea de la forma $8n + 7$ (y lo mismo puede decirse para los primos de la forma $8n + 3$).

Las pautas para un algoritmo de generación de primos seguro en el actual desarrollo de la teoría podrían ser:

- Verificar por tentativa si el número n candidato a primo es divisible por los números primos hasta 65536 para lo que se generará una tabla con tales números.

- Si n no es divisible por ningún primo menor que 65536, entonces le aplicamos el Test de Rabin-Miller un cierto número de veces. Para números n de más de 512 bits, aplicando el test 6 veces se tiene una probabilidad de error (que el número sea declarado primo siendo compuesto) menor a $2^{-70} < 10^{-20}$.

2 Firmas Electrónicas

Al realizar una transacción electrónica por computadora, si los montos de dinero que involucran son muy grandes deseamos tener la seguridad de que aquellos a quienes compramos o vendemos firmen la transacción haciéndose responsables de la misma. Esto en los documentos escritos en papel no presenta dudas de como se hace, mas allá de los falsificadores de firmas. En ultima instancia un perito calígrafo deberá intervenir en caso de dudas. ¿Cómo hacer en el caso de un documento electrónico, que está en un soporte magnético como un diskette o en un CD o en la memoria de un ordenador?
Una solución a este problema la da el sistema RSA como sigue: Al finalizar el documento firmamos un bloque de datos que contara con la información del remitente, la fecha y hora de la transacción, las características del documento, etc. Supongamos que ese bloque es M y que el remitente A tenga clave publica N_A, e_A y que su clave privada es d_A. Para firmar el documento A aplica su clave privada d_A, **que solo el conoce**, a M obteniendo $C = M^{d_A} \ mod \ N_A$. En principio C es incomprensible. Pero si se le aplica la clave publica e_A a C se obtiene

$$C^{e_A} = (M^{d_A})^{e_A}) \ mod \ N_A = M$$

Así el mensaje incomprensible adquiere sentido. Esto sólo es posible si el que lo envió posee a d_A. Como solo A conoce su clave privada, consideramos que esto es una prueba de que quien envió el documento realmente es quien dice ser.

3 Logaritmo Discreto

Cuando trabajamos con números reales la exponenciación no es mucho mas sencilla que la logaritmación, ie: calcular $b^x = y$ no es mas simple que calcular $\log_b y = x$. Pero supongamos que tenemos un grupo finito G como por ejemplo $\mathbb{Z}_p \backslash \{0\}$ con el producto módulo p, p primo. El algoritmo de exponenciación rapida permite calcular $b^x \ mod \ p$ en forma bastante eficiente (en tiempo polinomial en la longitud de x). Pero si conocemos que $y \in G$ es de la forma b^x, $b \in G$, $x \in \mathbb{Z}$, cómo hallar un x tal que $b^x = y$ no es tan simple. Se conjetura que no existe, en el caso general, un algoritmo que resuelva este problema de manera eficiente (polinomial

en las longitudes de las entradas y y b). Este problema, el de dados y y b hallar x, se conoce como el problema del logaritmo discreto.

Definición 3.0.1 *Si G es un grupo finito y $b, y \in G$ y y es una potencia de b, el logaritmo discreto de y respecto a la base b es cualquier entero x tal que $b^x = y$.*

Ejemplo:
Si tomamos $G = \mathbb{Z}_{19}^*$ con el producto módulo 19, entonces 2 es un generador de G y $\log_2 7 = 6$ pues $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 13, 2^6 = 7$, todo módulo 19.

Si G es un grupo de orden primo, entonces es cíclico y cualquier elemento $g \in G$, $g \neq e$ es un generador de G. Aquí e es la identidad de G. En este caso siempre existe el logaritmo discreto respecto a la base g. Ser9a pues bueno poder aseverar que G es de orden primo.
Si $G = \mathbb{Z}_p$, p primo, con la suma de enteros módulo p, g es un generador si $g \neq 0$. En este caso g^x es gx y entonces $y = g^x = xg = g+g+\cdots+g$ (x veces), todo módulo p. Pero en este caso el Algoritmo de Euclides Extendido nos permite, conocidos g y y resolver fácilmente $gx \equiv y \ mod \ p$. Así que este caso no nos sirve, así como no sirve ningún caso en que sea fácil calcular "el logaritmo discreto". Si tomamos $G = \mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\}$, con el producto módulo p, en ese caso no resulta fácil calcular x a partir de $g^x \equiv y \ mod \ p$, pero el grupo G tiene $p-1$ elementos que no es un número primo. Tampoco es fácil hallar un generador g de $\mathbb{Z}_p^*$. En general nos conformamos con encontrar un g tal que el subgrupo generado por él,

$$< g >= \{1, g, g^2, g^3, \ldots, g^n\}$$

tenga un orden grande.

3.1 Sistemas de encriptar basados en el logaritmo discreto

3.1.1 Diffie-Hellman

Dado que los sistemas de clave pública son más lentos que los de clave simétrica, Diffie y Hellman desarrollaron un sistema basado en el logaritmo discreto para intercambiar claves a usar en un sistema simétrico.

Supongamos que el sistema de encriptar a usar en el sistema simétrico utiliza cierta clave k que eligen al azar. Bernarda Alba y Antonio Torres deben ponerse de acuerdo en dicha clave sin transmitírsela a ningún posible espía. Para eso eligen un cuerpo finito que supondremos que es en nuestro caso $\mathbb{Z}_p$ con p primo muy grande, digamos, por ejemplo, de unos 1024 bits. En $\mathbb{Z}_p^*$ con el producto módulo p, elegimos un elemento g, tal que el subgrupo generado por g sea de un orden grande, idealmente un generador de $\mathbb{Z}_p^*$. Tanto g como $\mathbb{Z}_p$ son públicos.

Antonio elige al azar un número a entre 1 y $p-1$ que es mantenido en secreto, y calcula $g^a \ mod \ p$ y se lo envía a Bernarda. Bernarda a su vez elige al azar un número b entre 1 y $p-1$ que también guarda en secreto y calcula $g^a \ mod \ p$ y se lo envía a Antonio. La clave que van a usar Antonio y Bernarda en el sistema simétrico es g^{ab} (o parte de ella). Antonio puede calcularla pues ha recibido g^b de Bernarda y conociendo a que él mantuvo en secreto, calcula g^{ab} como $g^{ab} = (g^b)^a$. Por su lado Bernarda que recibió g^a enviado por Antonio y conoce b, calcula g^{ab} como $g^{ab} = (g^a)^b$. Un posible espía puede conocer g^a y g^b si espió la transmisión de los mensajes entre Antonio y Bernarda, pero a partir de ellos NO puede calcular g^{ab} a menos (conjetura) que calcule a o b (resuelva el problema del logaritmo discreto). Observar que multiplicar g^a por g^b da g^{a+b} NO g^{ab}. El sistema de Diffie-Hellman es utilizado por varias empresas que otorgan certificados electrónicos.

3.1.2 ElGamal, sistema de encriptado y de firma electrónica

Como en 3.1.1, elegimos $\mathbb{Z}_p$ con p primo muy grande (o más generalmente algún cuerpo finito, F_q, $q = p^n$) y lo hacemos público, así como a un elemento g de orden muy grande, preferentemente un generador de $\mathbb{Z}_p^*$. Cada usuario Ulises elige *al azar* un número $a = a_U$ que mantiene secreto, y hace público a g^a que es la clave pública de U. De algún modo un mensaje se codifica como una sucesión de números en $\mathbb{Z}_p$, y hay acuerdo sobre esto entre los usuarios. Supongamos que M es el mensaje a enviarle a Ulises por parte de Romina. para ello Romina busca la clave pública g^a de Ulises en algún directorio *ad hoc* y al azar elige k y calcula g^k y $g^{ak} = (g^a)^k$. Luego envía a Ulises la pareja (g^k, Mg^{ak}). Ulises, que conoce a puede calcular $g^{ak} = (g^k)a$, y descifrar el mensaje multiplicando Mg^{ak} por $g^{-ak} = g^{(p-a)k}$.

Si alguien sabe calcular a a partir de g^a entonces puede romper el código y descifrar el mensaje.

Tanto el sistema RSA como el de ElGamal permiten firmar un mensaje electrónicamente. Veamos el método de ElGamal.

Son públicos un primo p y $g \in \mathbb{Z}_p^*$ preferentemente un generador de $\mathbb{Z}_p^*$.

Supongamos que la clave publica de A es $g^a \ mod \ p$ y su clave privada es a entre 1 y $p-1$. Para firmar A el mensaje m hace lo siguiente:

1. Elige al azar k tal que $MCD(k, p-1) = 1$. El valor de k debe mantenerlo secreto A.
2. Calcula $r = g^k \ mod \ p$
3. Resuelve la ecuación en t: $m = ar + kt \ mod \ p-1$ usando el AEE.
4. Envía (m, r, t).

El receptor del mensaje, digamos B, calcula $g^m \ mod \ p$ y comprueba que es congruente con $(g^a)^r r^t \ mod \ p$ lo que le garantiza que el mensaje es de A.

En efecto como $m \equiv ar + kt \ mod \ p-1$ por lo que $g^m \equiv g^{ar+kt} = (g^a)^r (g^k)^t = (g^a)^r r^t \ mod \ p$ ya que por el Teorema de Fermat $g^{p-1} \equiv 1 \ mod \ p$.

¿Por qué está seguro B de que el mensaje es de A? Un impostor debería poder calcular r y t a partir de m y de g^a. Puede fácilmente hallar g^m y conoce la clave pública de A, $H = g^a$. Si desea generar una firma "creíble"debe hallar r y t que verifiquen $g^m \equiv (g^a)^r r^t \ mod \ p$. Se conjetura que esto no es posible hacerlo sin resolver el problema del logaritmo discreto. Si este problema se sabe resolver entonces de g^a se sabe calcular a y firmar un espía cualquier mensaje haciéndose pasar por A.

El valor de k debe calcularse cada vez *al azar*, de divulgarse se revela también la clave secreta a, pues si se conoce k entonces de $m = ar + kt \ mod \ p-1$ se conoce todo menos a que puede calcularse usando el AEE.

Tampoco debe usarse NUNCA el mismo valor de k para firmar dos mensajes. Si no un espía podría recuperar el valor de la clave privada a.

En efecto: Si tenemos como firmas las ternas (r, t_1, m_1), $(r, t_2.m_2)$, con $r = g^k$, entonces

$$(g^a)^r r^{t_1} \equiv g^{m_1} \ mod \ p \quad (g^a)^r r^{t_2} \equiv g^{m_2} \ mod \ p$$

lo que implica que

$$r^{t_2 - t_1} \equiv g^{m_2 - m_1} \ mod \ p$$

Como $r = g^k$ se tiene que

$$k(t_2 - t_1) \equiv m_2 - m_1 \; mod \, p - 1$$

Si $t_2 - t_1$ es invertible módulo $p-1$ resulta inmediatamente el valor de k y de aquí, como antes hallamos a. Si no es invertible y $MCD(t_2 - t_1, p-1) = d > 1$ resulta que $m_2 - m_1$ es múltiplo de d(pues existe solución) y se obtiene

$$k\frac{(t_2 - t_1)}{d} \equiv \frac{m_2 - m_1}{d} \; mod \, \frac{p-1}{d}$$

Esto da varias opciones para k. Si k_1 es una solución módulo $\frac{p-1}{d}$ entonces entre los d valores

$$k_1, \, k_1 + \frac{p-1}{d}, \, k_1 + 2\frac{p-1}{d}, \ldots, k_1 + (d-1)\frac{p-1}{d}$$

está la solución módulo $p-1$ para k.

3.1.3 Cálculo de logaritmos discretos

Supongamos que todos los factores primos de $p-1$ son pequeños, ie: menores que un cierto K. Por ejemplo si $p = 2053$ entonces $p - 1 = 2052 = 2^2 \times 3^3 \times 19$. Aquí podemos tomar $K = 20$. Cuando los factores primos de $p-1$ son chicos se dice que $p-1$ es suave.

Asumiendo esto para $p-1$ existe un algoritmo rápido para calcular el logaritmo discreto respecto a la base b.

Primero intentamos factorizar $p-1$. Como suponemos $p-1$ suave vamos a tener éxito y escribimos $p - 1 = q_1^{h_1} \cdots q_k^{h_k}$ con q_j primo para cada $j = 1, \ldots k$, $q_j \neq q_l$ si $j \neq l$. Para cada q_j computamos las q_j raíces de la unidad de b en $\mathbb{Z}_p$ dadas por

$$r_{q_j,i} = b^{i(p-1)/q_j} \; mod \, p \;\; i = 0, 1, \ldots q_j - 1$$

Con esto tenemos la siguiente "tabla de logaritmos"para cada primo q_j

$$\begin{array}{cccccc}
r_{q_1,0} & r_{q_1,1} & \cdots & r_{q_1,q_1-1} & & \\
r_{q_2,0} & r_{q_2,1} & \cdots & \cdots & r_{q_2,q_2-1} & \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
r_{q_k,0} & r_{q_k,1} & \cdots & \cdots & \cdots & r_{q_k,q_k-1}
\end{array}$$

Si $p-1 = \prod_j q_j^{h_j}$, para hallar el logaritmo discreto x de $y = b^x$ respecto a b alcanza con hacerlo respecto a $q_j^{h_j}$ para cada j y luego aplicar el Teorema Chino del Resto. Entonces fijemos $q = q_j$ tal que $q^{h_j} = q^h$ divide exactamente a $p-1$ y busquemos como determinar $x \ mod \ q^h$. Para eso escribimos $x = x_0 + x_1 q + \cdots + x_{h-1} q^{h-1} \ mod \ q^h$ con $0 \leq x_i < q$. Para hallar x_0 calculamos $y^{(p-1)/q} \ mod \ p$. Obtenemos una q-ésima raíz de la unidad módulo p pues $y^{q-1} \equiv 1 \ mod \ p$. Como

$$y = b^x = b^{x_0 + x_1 q + \cdots + x_{h-1} q^{h-1}}$$

queda

$$y^{(p-1)/q} = b^{(x_0 + x_1 q + \cdots + x_{h-1} q^{h-1})(p-1)/q} =$$

$$= b^{x_0 (p-1)/q} \ mod \ p$$

pues los términos $x_j q^j$ del exponente con $j \geq 1$ cancelan con el q que divide quedando $b^{(p-1)H}$ para algún h entero, y esto es congruente con 1 módulo p por el Teorema de Fermat. Se tiene que

$$y^{(p-1)/q} \equiv b^{x(p-1)/q} \equiv b^{x_0 (p-1)/q} \equiv r_{q,x_0} \ mod \ p$$

Comparando en la "tabla de logaritmos" cuál valor de i cumple $r_{q,x_0} = r_{q,i}$ ponemos $x_0 = i$.

Para hallar x_1 reemplazamos y por $y_1 = y / b^{x_0} \ mod \ p$. Queda $y_1 = b^{x_1 q + \cdots + x_{h-1} q^{h-1}}$. Elevando y_1 a la potencia $(p-1)/q^2$ módulo p obtenemos

$$y_1^{(p-1)/q^2} \equiv b^{(x_1 q + \cdots + x_{h-1} q^{h-1})(p-1)/q^2} \equiv$$

$$b^{x_1 (p-1)/q} \equiv r_{q,x_1} \ mod \ p$$

Otra vez comparamos r_{q,x_1} con los valores de $r_{q,i}$ de la tabla y cuando $r_{q,x_1} = r_{q,i}$ hacemos $x_1 = i$.

Por inducción, cuando ya calculamos $x_0, x_1, \ldots, x_{r-1}$ calculamos x_r haciendo primero

$$y_r = y_{r-1} / b^{x_{r-1} q^{r-1}}$$

y luego elevando y_r a la potencia $(p-1)/q^{r+1}$. El resultado va a dar

$$y_r^{(p-1)/q^{r+1}} \equiv b^{x_r (p-1)/q} \equiv r_{q,x_r} \ mod \ p$$

y hacemos $x_r = i$ si $r_{q,x_r} = r_{q,i}$. Al finalizar (cuando $r = h-1$) tendremos $x \ mod \ q^h$. Haciendo esto para todo q primo divisor de $p-1$ y aplicando el Teorema Chino del Resto hallamos un $x \ mod \ p-1$ tal que $b^x \equiv y \ mod \ p$.

Ejemplo: Tomemos $p = 2053$ entonces $p - 1 = 2052 = 2^2 \times 3^3 \times 19$.

$$r_{2,0} = 1, \ \ r_{2,1} = 2^{(p-1)/2} = 2^{1026} = -1 = 2052 \ mod \ 2053$$

$$r_{3,0} = 1, \ \ r_{3,1} = 2^{(p-1)/3} = 2^{684} = 1855 \ mod \ 2053, \ \ r_{3,2} = 2^{2(p-1)/3} = (2^{684})^2 = 197 \ mod \ 2053$$

$$r_{19,0} = 1, \ \ r_{19,1} = 2^{(p-1)/19} = 2^{108} = 1929 \ mod \ 2053, \ \ r_{19,2} = 2^{2\times 108} = 1005 \ mod \ 2053$$

completamos la tabla, se sobreentiende que los cálculos son módulo $p = 2053$.

$$r_{19,3} = 613, \ \ r_{19,4} = 2002, \ \ r_{19,5} = 165, \ \ r_{19,6} = 70$$

$$r_{19,7} = 1585, \ \ r_{19,8} = 548, \ \ r_{19,9} = 1850, \ \ r_{19,10} = 536$$

$$r_{19,11} = 1285, \ \ r_{19,12} = 794, \ \ r_{19,13} = 88, \ \ r_{19,14} = 1406$$

$$r_{19,15} = 161, \ \ r_{19,16} = 566, \ \ r_{19,17} = 1671, \ \ r_{19,18} = 149$$

Queremos calcular el logaritmo discreto en base 2 de 597. O sea calcular x tal que $2^x \equiv 597 \ mod \ 2053$ Escribimos $x = x_0 + x_1 2 \ mod \ 2^2$. Calculamos $597^{(p-1)/2} = 597^{1026} \ mod \ 2053 = 1$ entonces $r_{2,x_0} = r_{2,0} = 1 \Longrightarrow x_0 = 0$ Reemplazamos $y = 597$ por $y_1 = 597/2^{x_0} = 597$ (pues $x_0 = 0$) y pasamos a calcular x_1. Para ello calculamos $597^{(p-1)/2^2} = 597^{513} \ mod \ 2053 = 1$ Se tiene entonces que $x_1 = 0$ y resulta que $x = x_0 + x_1 2 = 0 \ mod \ 4$

Luego calculamos $x = x_0 + x_1 3 + x_2 3^2 \ mod \ 3^3$ (usamos otra vez los símbolos x_0, x_1 pero no deberían generar confusión ya que ahora estamos calculando x módulo $3^3 = 27$). Calculamos $597^{(p-1)/3} = 597^{684} \ mod \ 2053 = 1$ Queda que $x_0 = 0$. Seguidamente calculamos x_1. Para ello hay que multiplicar y por el inverso de 2^{x_0}, que en este caso es 1 por lo que $y = y_1 = 597$, y con ese valor y_1 hallar $y_1^{(p-1)/3^2} \ mod \ 2053$. Queda $597^{228} = 197 \ mod \ 2053 = r_{3,2}$ Por lo tanto $x_1 = 2$. Para completar hay que hallar x_2. Para ello al valor de y_1 lo multiplicamos por el inverso de $2^{x_1 3} = 2^6 \ mod \ 2053$ que es 1636. Resulta $y_2 = y_1 \times 1636 = 597 \times 1636 = 1517 \ mod \ 2053$ $y_2^{(p-1)/3^3} = 1517^{76} = 1 \ mod \ 2053$. Queda pues que $x = x_0 + x_1 3 + x_2 3^2 = 6 \ mod \ 3^3$ Para completar hay que calcular $y^{(p-1)/19} \ mod \ 2053$. Resulta $597^{108} = 613 \ mod \ 2053$. Este número corresponde a

$r_{19,3}$. Así que $x = 3\ mod\ 19$
Finalmente resta resolver usando el Teorema Chino del Resto el sistema

$$\begin{aligned} x &\equiv 0 \quad mod\, 4 \\ x &\equiv 6 \quad mod\, 27 \\ x &\equiv 3 \quad mod\, 19 \end{aligned}$$

El resultado es 60 mod $(4 \times 27 \times 19)$ (recordar que $4 \times 27 \times 19 = 2052$). Se puede comprobar que $2^{60} \equiv 597\ mod\ 2053$.

Si en la descomposición factorial de $p - 1$ hay un factor primo q "grande", no es posible construir una "tabla de logaritmos"como la anterior y, en general, no se sabe resolver el problema del logaritmo discreto.

4 Curvas Elípticas

4.1 Aspectos básicos

La teoría de las curvas elípticas sobre cuerpos finitos hoy en día es vista como una de las fuentes más adecuadas para fundar sistemas de criptografía de clave pública. La razón para ello es que las curvas elípticas sobre cuerpos finitos nos brindan una fuente gigantesca de grupos abelianos que aun siendo de orden grande permiten calcular en tiempo polinomial. Los cuerpos finitos F, sacándoles el 0, proporcionan grupos abelianos respecto a la multiplicación que pueden ser usados en Criptografía en sistemas basados en el problema del logaritmo discreto, pero dado un número N hay un cuerpo finito de N elementos si y solo si $N = p^n$ donde p es un número primo. Y si $N = p^n$, entonces solo existe un cuerpo F con ese número de elementos (a menos de un isomorfismo de cuerpos). Eso da rigidez a la elección de dichos cuerpos. Si por ejemplo uno elige $F = \mathbb{Z}_p$, elección por la que hemos optado en la sección anterior, solo existe un cuerpo de p elementos. Se corre el riesgo de que, siendo públicos p y la base g, un criptoanalista tenga tiempo suficiente como para realizar una "tabla de logaritmos"y de ese modo ponga en peligro la seguridad del sistema de encriptar obligando a cambiar de cuerpo $\mathbb{Z}_p$, ie: de primo p. En los sistemas basados en curvas elípticas se tiene mayor flexibilidad en la elección del grupo.
¿Qué es una curva elíptica?

En general es el conjunto de valores (x, y) que satisfacen a una ecuación de la forma

$$y^2 + a_{11}xy + a_{01}y = x^3 + a_{20}x^2 + a_{10}x + a_{00} \quad \text{(I)}$$

Aquí (x, y) son una pareja de elementos del cuerpo de base K que puede ser $\mathbb{R}$, los complejos, o cualquier otro cuerpo, en particular un cuerpo finito F_q donde $q = p^n$ indica el número de elementos de F_q.
Si además el cuerpo F_q es tal que en él vale $1 + 1 \neq 0$ y $1 + 1 + 1 \neq 0$ (ie: su característica no es ni 2 ni 3, lo que en el caso de un cuerpo finito F_q equivale a que su número de elementos $q = p^n$ cumple que $p \neq 2$, $p \neq 3$) entonces esta ecuación se puede llevar mediante cambios de variable a la forma

$$Y^2 = X^3 + aX + b \qquad \text{(II)}$$

En efecto: haciendo el cambio de variable $y = Y + hx + k$ en (I) nos queda

$$(Y + hx + k)^2 + a_{11}x(Y + hx + k) + a_{01}(Y + hx + k) = x^3 + a_{20}x^2 + a_{10}x + a_{00}$$

desarrollando aparecen términos

$$(Y + hx + k)^2 = Y^2 + 2hxY + 2kY + h^2x^2 + 2hkx + k^2$$

como $1+1 = 2 \neq 0$ podemos elegir h y k para lograr que $a_{11}+2h = 0$ y $a_{10}+2k = 0$ lo que permite eliminar los términos en xy y en y. Si ahora realizamos el cambio $x = X + l$ obtenemos que $x^3 = (X + l)^3 = X^3 + 3lX^2 + \cdots$ y, dado que $1+1+1 = 3 \neq 0$, podemos elegir l tal que $3l + a_{20} = 0$. Se concluye que podemos asumir en estos casos que directamente la ecuación es de la forma $y^2 = x3 + ax + b$.

Supondremos aquí que *siempre* en los cuerpos que trabajamos $2 = 1 + 1 \neq 0$ y $3 = 1 + 1 + 1 \neq 0$. Esto es cierto si, por ejemplo, el cuerpo es $\mathbb{Z}_p$ con $p > 3$ primo o es $\mathbb{R}$ ó $\mathcal{C}$ ó $\mathbb{Q}$.

Es necesario pedir algo más sobre la ecuación $y^2 = x3 + ax + b$, y para ello comenzaremos ejemplificando con los números reales. La ecuación $y^2 = x3+ax+b$ puede verse como la ecuación implícita $F(x, y) = y^2 - x^3 - ax - b = 0$. Si el punto (x_0, y_0) satisface la ecuación, entonces (x_0, y_0) es un valor regular (o no singular) de $F(x, y) = 0$ si el gradiente de $F(x, y)$ en (x_0, y_0) no se anula,, o sea, si al menos una de las derivadas parciales en el punto (x_0, y_0) es distinta de 0. En este caso

puede despejarse una variable (la x o la y) en función de la otra obteniendo una curva diferenciable en un entorno del punto (x_0, y_0) como expresión de la ecuación.

Vamos a pedir que todos los valores (x, y) que cumplen $F(x, y) = 0$ (o sea, $y^2 = x3 + ax + b$) sean regulares. Esto equivale a que no son nulas a la vez $2y = \frac{\partial F(x,y)}{\partial y}$ y $3x^2 + a = \frac{\partial F(x,y)}{\partial x}$. Pero si $2y = 0$ como $2 \neq 0$ queda que $y = 0$ lo que implica $y^2 = 0$, o sea, $y^2 = x^3 + ax + b = 0$.

Quiere decir que los puntos singulares se dan cuando a la vez $x^3 + ax + b = 0$ y su derivada $3x^2 + a = 0$. Un polinomio $P(x)$ y su derivada $P'(x)$ se anulan a la vez si y solo si $P(x)$ tiene raíces múltiples (en los complejos en el caso general de un cuerpo de característica 0) y esto si y solo si el máximo común divisor de $P(x)$ y $P'(x)$ no es un polinomio constante .

Analicemos cuándo $P(x)$ y $P'(x)$ no tienen factores comunes no constantes. En ese caso su máximo común divisor es 1, $MCD(P(x), P'(x)) = 1$. Recordar que $MCD(a(x), b(x)) = MCD(a(x) + f(x)b(x), b(x))$, aplicando esto reiteradamente obtenemos:

$$MCD(x^3 + ax + b, 3x^2 + a) = MCD(x^3 + ax + b - \frac{1}{3}x(3x^2 + a), 3x^2 + a) =$$

supongamos momentáneamente que $a \neq 0$

$$MCD(= \frac{2}{3}ax + b, 3x^2 + a) = MCD(\frac{2}{3}ax + b, 3x^2 + a - \frac{9x}{2a}(\frac{2}{3}ax + b)) =$$

$$MCD(\frac{2}{3}ax + b, -\frac{9b}{2a}x + a)$$

Estos dos polinomios son de grado 1, no tienen factores comunes si y solo si no tienen la misma raíz. La raíz del primer polinomio es $x = -(3b)/(2a)$. Sustituyendo en el segundo polinomio debe dar una expresión distinta de cero al no tener raíces comunes, o sea

$$-\frac{9b}{2a}(-\frac{3b}{2a}) + a) = \frac{27b^2}{4a^2} + a \neq 0$$

Multiplicando por $4a^2$ obtenemos

$$\Delta = \Delta(a, b) = 27b^2 + 4a^3 \neq 0$$

La expresión anterior se llama el discriminante de $P(x) = x^3 + ax + b$. Si Δ no se anula no tiene $P(x)$ raíces múltiples lo que en nuestro caso equivale a que la curva

representada por $y^2 = x^3 + ax + b$ no tiene puntos singulares. El caso $a = 0$ queda incluído en el anterior: si $a = 0$ resulta $x^3 + ax + b = x^3 + b$ y su derivada es $3x^2$. La anulación de la derivada se da entonces si $x = 0$. Si la derivada no tiene raíces comunes con $x^3 + b = 0$, entonces debe ser $b \neq 0$ y $\Delta = 27b^2 \neq 0$.

Lo anterior se extiende sin dificultad a cualquier cuerpo (finito o no) definiendo la derivada de polinomios de modo formal. Si

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

entonces $P'(x)$ es

$$P'(x) = n a_n x n - 1 + (n-1) a_{n-1} x^{n-2} + \cdots 2 a_2 x + a_1$$

Esto se extiende a polinomios en varias variables del modo obvio. Resulta que la condición para que $x^3 + ax + b$ no tenga raíces múltiples es que $P(x)$ y $P'(x)$ no tengan factores comunes no constantes y esto implica (estamos suponiendo que $2 = 1 + 1 \neq 0$ y $3 = 1 + 1 + 1 \neq 0$) que $\Delta = 27b^2 + 4a^3 \neq 0$.

4.2 Suma de puntos sobre una curva elíptica

Los puntos sobre una curva elíptica forman un grupo abeliano respecto a la suma que vamos a introducir en forma geométrica. Asumamos por el momento que estamos en el caso real, $(x, y) \in \mathbb{R}^2$. Adjuntamos a la curva elíptica $\mathcal{E}$ un punto al "l infinito"O que actuará como neutro de la suma a definir.

- Observamos que si $P = (x, y) \in \mathcal{E}$ entonces de la simetría de la ecuación $y^2 = x^3 + ax + b$, resulta que $P' = (x, -y) \in \mathcal{E}$. Llamaremos a $P' = (x, -y)$ el opuesto de $P = (x, y)$ y lo denotaremos por $-P$.

- Para sumar dos puntos P y Q de $\mathcal{E}$ trazamos la recta PQ que une los puntos. Dado que la ecuación de la curva es de tercer grado, genéricamente va a volver a cortar a la curva elíptica $\mathcal{E}$ en un tercer punto R. También $-R$ está en la curva $\mathcal{E}$. Definimos $P + Q$ como el punto $-R$.

- Si P y Q son puntos de igual abscisa pero distinta ordenada entonces $Q = -P$ por nuestra definición anterior y ponemos $P + Q = O$, el "punto al infinito". Esto dice, en nuestra interpretación geométrica, que el "punto al infinito"hay que pensarlo como infinitamente alejado en la dirección del eje Oy.

- Si la recta PQ tiene una tangente en Q entonces $R = Q$ y ponemos $P + Q = -Q$
- Si $P = Q$ entonces sustituímos la recta secante en P y Q por la recta tangente por P a la curva $\mathcal{E}$. Como no hay puntos singulares, pues $\Delta \neq 0$, esta tangente existe. Si la tangente corta en un tercer punto R a $\mathcal{E}$, simetrizamos R respecto al eje Ox obteniendo $-R$ que es por definición la suma $P + P = 2P$.
- Puede ocurrir que la recta tangente tenga una tangencia de orden dos en P (si P es un punto de inflexión sobre la curva), en ese caso $P = Q = R$ y definimos $P + P = 2P = -P$ El gráfico de la figura ilustra el caso de la curva elíptica dada por la ecuación $y^2 = x^3 - x$

Figura 1: Ilustración de la suma de puntos

Nos interesa ver que efectivamente existe un punto más en la recta PQ y calcular de ese modo las coordenadas de la suma $P + Q$.
Consideremos el caso en que P y Q tienen diferentes abscisas: $P = (x_1, y_1)$, $Q = (x_2, y_2)$, $x_1 \neq x_2$. Si las abscisas coinciden definimos $P+Q$ como el punto al infinito O. La recta que pasa por P y Q tiene ecuación $y = mx + n$ (pues como $x_1 \neq x_2$ no es vertical) donde $m = \frac{y_2 - y_1}{x_2 - x_1}$, $n = y_1 - mx_1$. Un punto en la recta PQ está en la curva si

$$(mx + n)^2 = x^3 + ax + b$$

desarrollando la expresión y pasando todo a un miembro queda

$$x^3 - m^2x^2 + (a - 2mn)x + b - n^2 = 0$$

Esta ecuación siempre tiene tres raíces en el cuerpo en que estemos trabajando ya que siendo de tercer grado tiene al menos 2 raíces que son x_1, x_2 las abscisas de P y Q respectivamente, y la suma de las raíces es $x_1 + x_2 + x_3 = m^2$ de donde

$$x_3 = m^2 - x_1 - x_2 = \frac{(y_2 - y_1)^2}{(x_2 - x_1)^2} - x_1 - x_2$$

es la tercera raíz. Conociendo a x_3 obtenemos $\hat{y}_3$, la ordenada del punto de corte, como

$$\hat{y}_3 = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_1)$$

El punto suma de $P + Q$ es $-R = (x_3, -\hat{y}_3) =_{Not} (x_3, y_3)$ por lo que en definitiva las coordenadas de la suma quedan

$$x_3 = \frac{(y_2 - y_1)^2}{(x_2 - x_1)^2} - x_1 - x_2$$

$$y_3 = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3)$$

Si $P = Q$ consideramos la recta tangente en $P = (x_1, y_1)$ a la curva elíptica dada por $F(x, y) = y^2 - x^3 - ax - b = 0$. La ecuación de la pendiente m de dicha tangente es (aplicando la fórmula de la derivada de una función implícita)

$$m = \frac{3x_1^2 + a}{2y_1}$$

Al sustituir en la ecuación $y^2 = x^3 + ax + b$ da como antes

$$x^3 - m^2x^2 + (a - 2mn)x + b - n^2 = 0 \quad ((X))$$

donde n satisface

$$n = y_1 - mx_1 = y_1 - \frac{3x_1^2 + a}{2y_1}x_1$$

Ahora x_1 es raíz doble de $((X))$ por lo que

$$x_3 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1$$

La coordenada y_3 de la suma $P + P = 2P$ es

$$y_3 = -y_1 + \frac{3x_1^2 + a}{2y_1}(x_1 - x_3)$$

Estas fórmulas valen siempre que estemos en un cuerpo de característica distinta de 2 (no se cumpla $2 = 1 + 1 = 0$). La prueba de que estas fórmulas convierten a la curva elíptica con esta suma en un grupo abeliano será dada en un apéndice.
Ejemplo: Consideremos la curva elíptica de ecuación $y^2 = x^3 - 24x + 44$ No tiene

puntos singulares si $p > 7$ pues $\Delta = 27b^2 + 4a^3 = 27 \times 1936 - 4 \times 13824 = -3024 = -2^4 \times 3^3 \times 7$.

Calculemos la suma de $P = (2, 2)$ con $Q = (5, 7)$ que son puntos de la curva.

$$x_3 = \frac{(y_2 - y_1)^2}{(x_2 - x_1)^2} - x_1 - x_2 = \frac{25}{9} - 7 = -38/9$$

$$y_3 = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3) = -2 + \frac{5}{3}(2 + 38/9) = 226/27$$

Hallemos $P + P = 2P$.

$$x_3 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1 = (\frac{-12}{4})^2 - 4 = 5$$

$$y_3 = -y_1 + \frac{3x_1^2 + a}{2y_1}(x_1 - x_3) = -2 + \frac{-12}{4}(-3) = 7$$

En este caso particular queda $2P = Q$ por lo que $P + Q = 3P$.

¿Cómo puede formalizarse el "punto al infinito"O? Un modo para hacerlo es considerar el plano proyectivo que puede representarse como formado por las clases de equivalencia de puntos $(X, Y, Z) \neq (0, 0, 0)$, con $X, Y, Z \in F$ con F el cuerpo en que estemos trabajando, bajo la relación de equivalencia $(X, Y, Z) \sim (X', Y', Z')$ si existe $\lambda \in F$ tal que $(X', Y', Z') = (\lambda X, \lambda Y, \lambda Z)$. Una clase de equivalencia es lo que se llama un punto en el plano proyectivo. Los puntos para lo que $Z \neq 0$ se llaman puntos propios y tienen un representante de la forma $(x, y, 1)$ basta hacer $x = X/Z$, $y = Y/Z$. Si $Z = 0$ se tienen los puntos impropios que pueden verse como las direcciones asintóticas al alejarse sobre una recta hacia el infinito, pueden verse como "el horizonte"del plano y constituyen lo que se llama la recta impropia del plano. Toda ecuación polinomial $F(x, y) = 0$ en el plano afín se transforma en una ecuación polinómica homogénea en el plano proyectivo de la forma $\hat{F}(X, Y, Z) = 0$. Basta sustituir x por X/Z y y por Y/Z y multiplicar por una potencia conveniente de Z para eliminar denominadores en la indeterminada Z. Para el caso de la ecuación

$$y^2 = x^3 + ax + b \quad (AA)$$

este procedimiento da

$$(\frac{Y}{Z})^2 = (\frac{X}{Z})^3 + a\frac{X}{Z} + b$$

Multiplicando por Z^3 queda la ecuación proyectiva

$$Y^2Z = X^3 + aXZ^2 + bZ^3 \quad (HH)$$

Todos los puntos de la forma (X, Y, Z) con $Z \neq 0$ que satisfacen (HH) es porque satisfacen, al dividir por Z, la ecuación (AA). Aparte de estos el punto con coordenada $Z = 0$ que satisface (HH) cumple $X^3 = 0$ lo que como F es un cuerpo implica que $X = 0$. Como no pueden anularse a la vez X, Y y Z resulta que $Y \neq 0$. Un representante de la clase de equivalencia de este punto es $(0, 1, 0)$. Esto coincide con la idea de que se trata de un punto infinitamente alejado en la dirección de Oy.

4.3 Curvas elípticas sobre cuerpos finitos

Nos restringimos al caso de $F = \mathbb{Z}_p$. Es trivial que si el cuerpo F tiene p elementos una cota del número de elementos en la curva elíptica $\mathcal{E}$ dada por $y^2 = x^3 + ax + b$ está dada por $2p+1$ (por cada x hay dos posibles valores de y que pueden satisfacer $y^2 = x^3 + ax + b$ y como hay p posibilidades para x esto da $2p$ puntos propios sobre $\mathcal{E}$, debemos sumarle el punto impropio O y esto da $2p+1$). Pero solo para la mitad de los elementos a de $F = \mathbb{Z}_p$ existe b tal que $b^2 = a$ y de aquí es razonable pensar que solo para la mitad de los valores $x^3 + ax + b$ va a existir tal y. Es de esperar entonces que existan del orden de p puntos sobre $\mathcal{E}$.

Mas exactamente: definamos $(a|p)$ el símbolo de Legendre de a respecto a p como 1 si existe b tal que $b^2 = a$ en $\mathbb{Z}_p$ y $MCD(a, p) = 1$, -1 si no existe tal b (en cuyo caso es claro que $MCD(a, p) = 1$) y 0 si a es múltiplo de p. Se cumple que $(a|p) = a^{(p-1)/2} \, mod \, p$.
En efecto si existe b tal que $b^2 = a$ entonces $a^{(p-1)/2} = (b^2)^{(p-1)/2} = b^{p-1} \equiv 1 \, mod \, p$ por el Teorema de Fermat. También $(a|p) = 1$. El polinomio $x^{(p-1)/2} = 1$ tiene a lo más $(p-1)/2$ en $\mathbb{Z}_p$ ya que éste es un cuerpo. Pero hay $(p-1)/2$ elementos en $\mathbb{Z}_p$ distintos entre sí que satisfacen esta relación, a saber $1 = (p-1)^2, 2^2 = (p-2)^2, 3^2 = (p-3)^2, \ldots, (\frac{p-1}{2})^2 = (\frac{p+1}{2})^2$. Pues si j, h están entre 1 y $(p-1)/2$ y $j^2 = h^2 \, mod \, p$ entonces $(j-h)(j+h) \equiv 0 \, mod \, p$ y como j, h están entre 1 y $(p-1)/2$ entonces $2 \leq j + h \leq p - 1$ y resulta que $j - h = \dot{p}$, o sea $j = h \, mod \, p$. Luego ninguno de los elementos a para los que no existe b tal que $b^2 = a$ pueden satisfacerla. Para estos a se cumple que $a^{(p-1)/2} \neq 1$. Pero como $a^{p-1} = 1$ entonces $z = a^{(p-1)/2}$ es raíz de $z^2 = 1$. Como en $\mathbb{Z}_p$, *por ser p primo*, solo -1 y 1 son

solución de $z^2 = 1$ queda que $a^{(p-1)/2} = -1$, por lo que coincide con $(a|p)$ en este caso también. El caso p divisor de a es trivial.

Con la definición del símbolo de Legendre se tiene que $1 + (x|p)$ da el número de soluciones de $y^2 = x$, $\mathbb{Z}_p \ni x \neq 0$ que son 2 si $(x|p) = 1$ 1 si $(x|p) = 0$ y 0 si $(x|p) = -1$. Entonces el número de puntos de $\mathcal{E}$ es

$$1 + \sum_{x \in \mathbb{Z}_p} (1 + (x^3 + ax + b|p)) = p + 1 + \sum_{x \in \mathbb{Z}_p} (x^3 + ax + b|p)$$

Es de esperar que $(x^3 + ax + b|p)$ tome en forma equitativa los valores 1 y -1 y que se comporte en cierto sentido como una caminata aleatoria con probabilidad $\frac{1}{2}$ de tomar uno u otro valor. En probabilidad se prueba que si esto fuera cierto entonces el valor esperado de $\sum_{x \in \mathbb{Z}_p} (x^3 + ax + b|p)$ es 0 y la desviación estándar es $\sqrt{p}$. En realidad se prueba que si F es un cuerpo finito de q elementos, vale que

Teorema 4.3.1 *(Teorema de Hasse) Sea N el número de F-puntos sobre la curva elíptica $\mathcal{E}$ definida sobre F. Entonces*

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

Además de conocer el número de puntos de $\mathcal{E}$ nos interesa saber la estructura de $\mathcal{E}$ como grupo abeliano. Este grupo no es necesariamente cíclico pero se puede probar que siempre es el producto de dos grupos cȯclicos. Eso significa que es isomorfo al producto de dos grupos de la forma $\mathbb{Z}_s \times \mathbb{Z}_r$ donde s, r son divisores de N.

Para contar el número de puntos N lo escribimos como $N = q + 1 - t$, por el teorema de Hasse $|t| \leq 2\sqrt{q}$. El número t se llama la traza de Frobenius de la curva elíptica $\mathcal{E}$.

Para determinar t puede hacerse módulo primos l menores o iguales que L tales que

$$M = \prod_{l \ primo,\ 2 \leq l \leq L} l > 4\sqrt{q}$$

A partir del Teorema Chino de los restos resulta t determinado módulo $M > t$ por lo que conocemos el verdadero valor de t.

Esta es la idea de un algoritmo debido a Schoof para la determinación de la traza de Frobenius t.

Por ejemplo, como $q+1$ es par resulta que $t = N \, mod \, 2$. Entonces t es impar si y solo si x^3+ax+b es irreducible sobre $F = \mathbb{Z}_p$. Esto porque de ser irreducible sobre F no tiene x^3+ax+b raíces en F y por lo tanto para cada $x \in F$ que cumpla que existe $y \in F$ tal que $y^2 = x^3+ax+b$ también $-y$ es solución y como $x^3+ax+b \neq 0$ resulta $y \neq -y$.

Luego el número de soluciones en este caso es el punto al infinito 0 más un número par, o sea 1 módulo 2.

Si por otra parte x^3+ax+b no es irreducible sobre F entonces al ser de tercer grado se factoriza como un polinomio de segundo grado por uno de primer grado, luego tiene una raíz en F. Si tiene una sola, x_1 entonces para este valor existe uno solo $y_1 = 0$ tal que $y_1^2 = x_1^3 + ax_1 + b = 0$. Todos los otros valores de x tal que existe y que cumple $y^2 = x^3+ax+b$ dan dos valores distintos y y $-y$. Pero si hubiera dos raíces en F, x_1, x_2 entonces de la relación entre coeficientes y suma de raíces de una ecuación polinómica aplicada a este caso particular sacamos que la tercera raíz x_3 es $-x_1 - x_2$ y también esta en F. Por otra parte, como x^3+ax+b no tiene raíces múltiples pues $\Delta = 27b^2 + 4a^3 \neq 0$ esto dice que siempre hay un número impar de raíces en F.

Luego el número de soluciones en este caso es el punto al infinito 0 más un número impar, o sea 0 módulo 2.

Si el primo l es mayor que 2 los argumentos son más complicados y utilizan lo que se llama el endomorfismo de Frobenius de la curva elíptica y los llamados polinomios de división. No entraremos en detalles contentándonos con calcular el orden de $\mathcal{E}$ en el siguiente ejemplo.

Ejemplo: Consideremos la curva elíptica $\mathcal{E}$ de ecuación $y^2 = x^3 - 24x + 44$ sobre $F = \mathbb{Z}_{59}$. El número N de elementos de $\mathcal{E}$, según el Teorema de Hasse cumple $|N-(q+1)| \leq 2\sqrt{q}$ o sea $|N-60| \leq 2\sqrt{59}$, lo que da que N varía entre 45 y 75.

Para ver si $x^3 - 24x + 44$ es irreducible sobre el cuerpo $\mathbb{Z}_{59}$ alcanza con ver si $MCD(x^3+ax+b, x^q - x) = 1$ Esto se debe a que $x^{59} - x$ contiene todas las raíces de $\mathbb{Z}_{59}$ (en el caso de que $F = \mathbb{Z}_p$ esto es esencialmente el Teorema de Fermat). En nuestro caso $MCD(x^3 - 24x + 44, x^{59} - x) = 1$ así que $t = 1 \, mod \, 2$. Esto reduce los casos de t posibles, en nuestro caso t es impar. Tomemos $P = (2,2) \in \mathcal{E}$, y calculemos el orden del subgrupo generado por P sobre $F =$

$\mathbb{Z}_{59}$, si éste es suficientemente grande, nos permitirá tener una estimación del orden del grupo $\mathcal{E}$, esta técnica es costosa computacionalmente, y el algoritmo de Schoof permite obviarla, aunque los algoritmos de cálculo de t también son onerosos computacionalmente.

Tenemos que $2P = (5, 7) \neq P$.

$$3P = (-38 \times 9^{-1} \, mod \, 59, 226 \times 27^{-1} \, mod \, 59) =$$

$$(21 \times 46 \, mod \, 59, 49 \times 35 \, mod \, 59) = (22, 4) \neq P$$

Calculemos $4P = 2P + 2P$

$$x_3 = ((3x_1^2 + a)(2y_1)^{-1})^2 - 2x_1 = ((51)(14)^{-1})^2 - 10 =$$

$$= ((51)(38))^2 - 10 = 22 - 10 = 12 \, mod \, 59$$

$$y_3 = -y_1 + (3x_1^2 + a)(2y_1)^{-1}(x_1 - x_3) = -7 + (51)38)(5 - 12) =$$

$$= -7 + 4 = -3 = 56 \, mod \, 59$$

Luego $4P = (12, -3) = (12, 56) \neq 3P$ en $\mathbb{Z}_{59}$. Computamos $5P = 4P + P = (12, -3) + (2, 2) = (1, 27) \, mod \, 59 \neq 3P$ Computamos $6P = 4P + 2P = (12, -3) + (5, 7)) = (32, 40) \, mod \, 59 \neq 3P$ Computamos $7P = 4P + 3P = (12, -3) + (22, 4)) = (19, 4) \, mod \, 59 \neq 3P$ Análogamente $8P = 4P + 4P = (57, 54) = (-2, -5) \, mod \, 59$

Comparamos ahora todos los elementos entre $8P$ y $15P$ con $7P$. Si en algún momento se cumple $mP = nP$ entonces $(m + j)P = (n + j)P$ para todo $j \geq 0$. No es difícil demostrar que comparando todos los elementos mP con m entre 2^k y $2^{k+1} - 1$ con el elemento $(2^k - 1)P$ vamos a hallar un valor de m para el que se cumpla la igualdad, esto permite liberar memoria en el algoritmo de comparación. En nuestro ejemplo esto no es problema práctico.

En el caso particular anterior, se tiene que $8P \neq 7P$. Seguidamente calculamos

$9P = 8P + P = (51, 45) \, mod \, 59 \neq (19, 4)$, $10P = (18, 55) \, mod \, 59 \neq 7P$, $11P = (29, 45) \, mod \, 59 \neq 7P$, $12P = (43, 7) \, mod \, 59 \neq 7P$, $13P = (23, 53) \, mod \, 59 \neq 7P$, $14P = (11, 52) \, mod \, 59 \neq 7P$, $15P = (47, 43) \, mod \, 59 \neq 7P$, $16P = (55, 31) \, mod \, 59 \neq 15P$.

Ahora solo duplicamos y comparamos con la lista anterior, la razón de hacer esto es que estamos seguros por el Teorema de Hasse de que el orden de la curva

está entre 45 y 75 por lo que es altamente probable que comparando $64P$ con la lista se encuentre una repetición.

$32P = (13, 10)\, mod\, 59$, $64P = (1, 32) = (1, -27) = -5P$. Entonces $69P = \mathcal{O}$, "el punto al infinito". Quiere decir que el orden de P es 69, 3 (que no es el caso), o 23. Si fuera 23 entonces $16P + 7P = \mathcal{O}$ lo que implicaría que $16P = -7P = (19, -4) = (19, 55)\, mod\, 59$, lo que en particular forzaría a que la abscisa de $16P$ fuera 19. Se concluye que 69, el orden de $P = (2, 2)$ es el de $\mathcal{E}$.

En general no es simple hallar el orden de $\mathcal{E}$. Diferentes métodos han sido desarrollados, entre ellos los de Shanks y Mestre (ver [SM]) y el ya mencionado de Schoof (ver [SCH]).

4.4 Codificación de textos como puntos sobre una curva elíptica

Dado un cuerpo finito F ($= \mathbb{Z}_p$ en nuestro caso) y una curva elíptica $\mathcal{E}$ de ecuación $y^2 = x^3 + ax + b$ sobre F queremos codificar nuestro texto como puntos (x, y) sobre $\mathcal{E}$. *Codificar no es encriptar.* No existe algoritmo que en tiempo polinomial realice esta tarea, pero si algoritmos probabilísticos que la realizan en tiempo polinomial.

Un posible método es el que sigue:

1. consideremos $\kappa \in \mathbb{N}$ tal que consideremos satisfactoria la probabilidad de que no podamos codificar un mensaje m si esta es del orden de $\frac{1}{2^\kappa}$. En la práctica $\kappa = 30$ o $\kappa = 50$ suelen bastar. Esto da una probabilidad del orden de $10-9$ y de 10^{-15} respectivamente.

2. si nuestra unidades de mensaje m son enteros $0 \leq m \leq M$ tomamos el primo $p > \kappa M$.

3. a m le asociamos el intervalo de κ números $\kappa m + j$ con $j = 1, 2, \ldots, \kappa$.

4. dado m tomamos $x = \kappa m + 1$ y calculamos el símbolo de Legendre $(x^3 + ax + b|p)$. Esto puede hacerse de modo eficiente usando que $(-1|p) = (-1)^{(p-1)/2}$; $(2|p) = (-1)^{(p^2-1)/8}$; si $a = bp + r$ entonces $((a|p) = (r|p)$; y si q es un impar existe una relación simple (ley de reciprocidad cuadrática) entre $(q|p)$ y $(p|q)$ que dice que ambos números coinciden salvo que ambos p y q sean congruentes con 3 módulo 4. (esto último debe verse como una generalización del símbolo de Legendre, llamada símbolo de Jacobi, ya que no asumimos que q sea

primo). En última instancia, como vimos más arriba siempre vale $(x^3+ax+b|p) = (x^3+ax+b)^{(p-1)/2} \, mod \, p$ que puede calcularse en tiempo polinomial. Si $(x^3+ax+b|p) = -1$ consideramos $x+1 = \kappa m + 2$ y reintentamos hasta hallar un x que cumpla $(x^3+ax+b|p) = 1$, asumiendo que lo encontramos antes de κ intentos, cosa que tiene una altísima probabilidad de ocurrir.

5. Una vez hallado x tal que $(x^3+ax+b|p) = 1$ hallamos y tal que $y^2 = x^3+ax+b$. Esto se puede lograr siempre que conozcamos algún a tal que para él $(a|p) = -1$. En el caso particular de que $p \equiv 3 \, mod \, 4$ este y se puede hallar como $(x^3+ax+b)^{(p+1)/4}$. Esto nos da el punto (x,y) en $\mathcal{E}$ que queremos.

6. Para recobrar m a partir de x calculamos

$$m = [\frac{x-1}{\kappa}]$$

4.5 Sistemas de Encriptar sobre Curvas Elípticas

El análogo de multiplicar dos elementos a, b en $\mathbb{Z}_N$ es el de sumar dos puntos P, Q sobre la curva elíptica $\mathcal{E}$. El análogo de calcular a^k en $\mathbb{Z}_N$ es el de multiplicar el entero k por el punto P de $\mathcal{E}$. El algoritmo para calcular potencias módulo N se traduce fácilmente a un algoritmo para multiplicar enteros k por puntos P. Primero escribimos k en la base 2, y luego usamos las fórmulas de duplicación de puntos. Si por ejemplo $k = 60$ lo escribimos como $57 = 32+16+8+1 = (111001)_2$. Ahora dado P calculamos con las fórmulas dadas anteriormente el valor de $2P = P+P$, $4P = 2P+2P$, $8P = 4P+4P$, etc. Y con las fórmulas de suma de puntos calculamos $57P = P+8P+16P+32P$, cada vez que encontramos un bit a 1 en la escritura de k en base 2, al valor inicial P le sumamos la potencia respectiva y acumulamos. Multiplicar por 0 es dar el punto al infinito $0P = \mathcal{O}$, el opuesto no presenta problemas: si $kP = (x,y)$ entonces $-kP = (x,-y)$.

Los sistemas criptográficos que discutiremos son análogos a los basados en el problema del logaritmo discreto sobre cuerpos finitos. En estos casos se fijaba un cuerpo F y un elemento g de $F\backslash\{0\}$ de orden grande, preferentemente un generador. Estos elementos se hacían públicos, conservando secreto cada usuario un número a elegido al azar y haciendo público g^a (por ejemplo en el sistema de ElGamal).

El problema análogo al del logaritmo discreto sobre curvas elípticas es el de dado dos puntos P y Q sobre $\mathcal{E}$ y sabiendo que $Q = aP$, hallar el valor de a. Este problema es tan intratable como el del logaritmo discreto al menos. En este caso fijamos una curva $\mathcal{E}$ y un punto base B que debemos chequear sea de un orden grande, éste oficiará de generador. Cambiando F por $\mathcal{E}$, g por B y la exponenciación en el cuerpo finito F por la multiplicación de enteros por puntos sobre la curva elíptica $\mathcal{E}$, pueden realizarse análogos de los métodos de cifrado de Diffie-Hellman, de Massey-Omura y de ElGamal. Los detalles quedan para el lector.

Uno de los problemas básicos aquí es elegir adecuadamente la curva elíptica y el punto base B. Nos referimos aquí solamente al caso de curvas sobre $\mathbb{Z}_p$ con p primo.

Uno de los procedimientos usuales pasa por elegir primero el primo $p > 3$ al azar (en general con un número conveniente de bits lo que para curvas el9pticas da no menos de 200 bits), luego elegimos, también al azar, x_0, y_0 y a en $\mathbb{Z}_p$ y hacemos $b = y_0^2 - x_0^3 - ax_0 \, mod \, p$. Luego chequeamos que $\Delta = 4a^3 + 27b^2 \neq 0 \, mod \, p$. Si esto no se cumple comenzamos una nueva elección al azar de x_0, y_0 y a. Hacemos $B = (x_0, y_0)$. Luego chequeamos de que el orden de B no sea bajo. Preferentemente que sea un generador de $\mathcal{E}$ y de que esta curva tenga un orden grande. Como comentamos antes, este problema es difícil computacionalmente y se utilizan para el varios métodos esbozados antes (ver bibliografía).

Referencias

[SM] COHEN H., *A Course in Computational Algebraic Number Theory,* Springer-Verlag, **GTM 138** 1993.

[SCH] SCHOOF, R., *Counting points on elliptic curves over finite fields,* J. Théorie des Nombres de Bordeaux, **7** (1995), p. 219-254.

[K] KOBLITZ, N., *A Course in Number Theorie and Cryptography,* Springer-Verlag, **GTM** 1987.

[SN] SCHNEIER, B., *Applied Cryptography,* John Wiley & Sons, Inc., **New York** 1996.

5 Ejercicios

1. Este ejercicio tiene como finalidad introducir una técnica de factorización conocida como Método de Fermat. Funciona con todo numero compuesto impar.

   (a) Demostrar que si N es impar existe una correspondencia biunívoca entre las formas de escribir $N = pq$ como producto de dos números y $N = t^2 - s^2$ como diferencia de dos cuadrados.

   (b) Probar que el siguiente algoritmo da la parte entera de $\sqrt{N}$, donde $N \geq 1$
   Paso 1: Definimos $X_0 = (N+1)/2$
   Paso 2: Dado X_k X_{k+1} se calcula como $X_{k+1} = \frac{X_k^2+N}{2X_k}$
   Paso 3: Si $X_{k+1}^2 - N < 1$ retornar el valor X_{k+1} sino volver al Paso 2

   (c) A partir de la parte entera de $\sqrt{N}$ intentar factorizar N impar usando que si $N = t^2 - s^2$ entonces si fijo un valor para t a partir de $[\sqrt{N}]+1$, $t = [\sqrt{N}]+1,\ [\sqrt{N}]+2, \ldots$ entonces obtendré una solución al problema (una factorización si $t^2 - N = s^2$ un cuadrado perfecto (esto puede chequearse con el algoritmo que da la parte entera de $\sqrt{t^2 - N}$).
   Ejemplo: Factorizar 200819. La parte entera de la raíz cuadrada de 200819 es 448. Tomando $t = 448 + 1 = 449$ se tiene que $t^2 - N = 449^2 - 200819 = 782$ que no es un cuadrado perfecto. Probando con $t = [\sqrt{200819}] + 2 = 448 + 2 = 450$ se tiene que $450^2 - 200819 = 1681 = 41^2$. Por lo tanto $200819 = 450^2 - 41^2$ de aquí que una factorización posible es $200819 = 491 \times 409$.

   (d) Con la ayuda de una calculadora para obtener la parte entera de la raíz cuadrada del número considerado, factorizar 8633, 809009, 88169891, 4601.

2. Este ejercicio tiene por finalidad mostrar un uso posible del Teorema de Fermat combinado con el Teorema Chino de los Restos.
   Calcular $2^{1000000}\, mod\, 77$
   Solución: $77 = 7 \times 11$, podemos calcular $2^{1000000}\, mod\, 7$ que da 2 pues por

el teorema de Fermat $2^6 \equiv 1 \, mod \, 7$ y $1000000 = 6 \times 166666 + 4$ por lo que $2^{1000000} \equiv 2^4 \, mod \, 7$ que es 2. Del mismo modo se obtiene que $2^{1000000} \equiv 1 \, mod \, 11$. Aplicando el Teorema Chino de los Restos se tiene que

$$2^{1000000} \equiv 23 \, mod \, 77$$

Hallar el último dígito de la representación de $2^{1000000}$ en base 13.

3. Resolver el sistema de congruencias

$$19x \equiv 103 \, mod \, 900$$

$$10x \equiv 511 \, mod \, 841$$

4. Hallar el menor entero positivo que dividido por 11 da resto 1, dividido por 12 da resto 2 y dividido por 13 da resto 3.

5. Resolver la ecuación $x^2 + 42x + 21 \equiv 0 \; mod \; 105$

6. Sea $p = 4k + 3$ un primo. Consideremos la ecuación $x^2 \equiv a \; mod \; p$
   Dar ejemplos de valores de a y p para los que la ecuación no tiene solución.
   Mostrar que si existe solución entonces las unicas soluciones b cumplen $b = \pm a^{p+1}$
   Escribir un programa que calcule las raíces cuadradas modulo p, la salida debe decir si existen raíces cuales son y si no decir que no hay solucion.

7. Supongamos que $N = pq$ y que $p \equiv q \equiv 3 \; mod \; 4$, $p \neq q$.
   Escriba un programa que calcule, conociendo p, q, a, las soluciones de $x^2 \equiv a \; mod \; N$. Sugerencia, use el Teorema Chino del Resto.

8. Usar el algoritmo de exponenciación veloz para hallar $38^{75} \, mod \, 103$

9. (Melange sobre congruencias, todo lo que quería saber pero le daba miedo preguntar).

   (a) Probar que si p es primo, $j \geq 2$ y tengo $p^s \, / \, h$, $p^s \, / \, k$, $0 < s < j$, la congruencia

   $$hf(x) + kg(x) \equiv 0 \; mod \; (p^j)$$

es equivalente a la congruencia

$$\frac{h}{p^s}f(x) + \frac{k}{p^s}g(x) \equiv 0 \; mod \; (p^{j-s})$$

(b) Sea $f(x) \equiv 0 \; mod \; (p^j)$ una congruencia y x_0 una solución de la misma. Probar que x_0 también es solución de $f(x) \equiv 0 \; mod \; (p)$

(c) (Método para resolver congruencias módulo p^j)

i. Dada la congruencia $F(x) \equiv G(x) \; mod \; (p^j)$ llévela del modo obvio a $f(x) \equiv 0 \; mod \; (p^j)$.

ii. Asumamos que $f(x)$ es un polinomio de grado $n > 0$

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0, \; a_j \in \mathbb{Z} \; \forall j$$

Pruebe que al calcular $\frac{1}{k!}f^{(k)}(x)$ queda un polinomio con coeficientes enteros. Aquí $f^{(k)}(x)$ es la derivada k-ésima de $f(x)$ (sí, la derivada). Así que la expresión tiene sentido módulo cualquier numero)

iii. Resuelva primero la congruencia $f(x) \equiv 0 \; mod \; (p)$

iv. Suponga que x_1 es una solución de $f(x) \equiv 0 \; mod \; (p)$. Pruebe usando la parte 9b de este ejercicio que si x es solución de $f(x) \equiv 0 \; mod \; (p^j)$ debe ser $x = x_1 + pt$, $t \in \mathbb{Z}$

v. Suponga que quiere resolver la congruencia $f(x) \equiv 0 \; mod \; (p^2)$ (ie: $j = 2$) Desarrolle $f(x)$ en el punto x_1 con incremento pt por la fórmula de Taylor (sí, de Taylor) obteniendo el valor en $x = x_1 + pt$. Va a quedar algo como

$$f(x) = f(x_1+pt) = f(x_1)+f'(x_1)pt+\frac{1}{2!}f''(x_1)(pt)^2+\cdots+\frac{1}{k!}f^{(k)}(x_1)(pt)^k+\cdots$$

Por la parte 9(c)ii $\frac{1}{k!}f^{(k)}(x_1) \in \mathbb{Z}$ así que $\frac{1}{k!}f^{(k)}(x_1)(pt)^k$ es múltiplo de p^k. Concluir que se pueden despreciar todos los términos del desarrollo de Taylor desde $k = 2$ en adelante y quedarse con

$$f(x) = f(x_1 + pt) = f(x_1) + f'(x_1)pt$$

para resolver $f(x) \equiv 0 \; mod \; (p^2)$.

vi. Suponga que $f'(x_1) \not\equiv 0 \ mod\ (p)$ ($f'(x_1)$ **no** es divisible por p). Recuerde que $f(x_1) \equiv 0 \ mod\ (p)$ ($f(x_1)$ **sí** es divisible por p) Use la parte 9a para llevar la congruencia módulo p^2

$$f(x) = f(x_1 + pt) = f(x_1) + f'(x_1)pt \equiv 0 \ mod\ (p^2)$$

a la congruencia (en la variable t) módulo p

$$\frac{f(x_1)}{p} + f'(x_1)t \equiv 0 \ mod\ (p)$$

Calculando t_1 solución, que existe y es unica pues supusimos que $f'(x_1) \not\equiv 0 \ mod\ (p)$, obtenemos $x_2 = x_1 + pt_1$ solución de $f(x) \equiv 0 \ mod\ (p^2)$

vii. Si quiere resolver una congruencia módulo p^3 y ya calculó x_2 solución de $f(x) \equiv 0 \ mod\ (p^2)$, por la parte 9b, una solución x de $f(x) \equiv 0 \ mod\ (p^3)$ se va a poder escribir $x = x_2 + p^2t$, con t algún entero. Reempiece de nuevo con el algoritmo: desarrolle por Taylor a $f(x)$ en el punto x_2. Observe que solo le importa la primera derivada (todo lo demás está multiplicado por potencias de p mas grandes que 3). Como $f'(x_2)$ no puede ser divisible por p si no lo era $f'(x_1)$ (¿por qué?) y en cambio $f(x_2)$ es divisible por p^2 la congruencia módulo p^3

$$f(x_2 + p^2t) = f(x_2) + f'(x_2)p^2t \equiv 0 \ mod\ (p^3)$$

se reduce a la congruencia módulo p

$$\frac{f(x_2)}{p^2} + f'(x_2)t \equiv 0 \ mod\ (p)$$

viii. Una nueva iteración de este procedimiento nos permite resolver $f(x) \equiv 0 \ mod\ (p^4)$, etc., etc.

10. (a) Use el ejercicio anterior para probar que si p es un primo impar, $x^2 \equiv 1 \ mod\ (p^j)$ solo tiene las soluciones 1 y $-1 = p^j - 1$.

(b) Pruebe que si $m = m_1 m_2 \cdots m_k$ con m_i, m_j primos entre si para $i \neq j$ resolver $x^2 \equiv 1 \ mod \ (m)$ es equivalente a resolver el sistema

$$\begin{array}{l} x^2 \equiv 1 \ mod \ (m_1) \\ x^2 \equiv 1 \ mod \ (m_2) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x^2 \equiv 1 \ mod \ (m_k) \end{array}$$

Demuestre que si $m = p_1^{j_1} p_2^{j_2} \cdots p_k^{j_k}$ con p_i primos impares distintos entre si, $i = 1, 2, \ldots, k$ entonces $x^2 \equiv 1 \ mod \ (m)$ tiene exactamente 2^k soluciones distintas. (O sea, el polinomio $x^2 - 1$ en $\mathbb{Z}_m$ aunque es de grado 2 tiene mas raíces que el grado.)

11. Demostrar que $ax^2 + bx + c \equiv 0 \ mod \ N$ con a invertible mod N es equivalente a $y^2 \equiv k \ mod \ N$.

12. Demostrar que si p es un numero primo entonces $x^2 \equiv a \ mod \ p$ tiene solución si y solo si $a^{(p-1)/2} \equiv 1 \ mod \ p$.

13. Demostrar que si p es un numero primo congruente con 3 mod 4 entonces si $x^2 \equiv a \ mod \ p$ tiene solución, esta puede hallarse como $b = a^{(p+1)/4} \ mod \ p$. Hay dos soluciones, la otra es $-b = p - b$.

14. Resolver

    (a) $x^2 \equiv 2 \ mod \ 121$

    (b) $x^2 + 14x + 46 \equiv 0 \ mod \ 121$

    (c) $x^2 \equiv 3 \ mod \ 103$

    (d) $x^2 \equiv 2 \ mod \ 1092727$

    (e) $x^2 \equiv 9 \ mod \ 12221$

15. Sea

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0, \ a_j \in \mathbb{Z}_p \ \forall j$$

un polinomio en $\mathbb{Z}_p$, $n > 0, a_n \neq 0 \ mod \ (p)$. Probar que siempre existe un polinomio $g(x)$ en $\mathbb{Z}_p$ de grado menor o igual que $p-1$ que tiene las mismas raíces que $f(x)$. [Sugerencias: Use el Teorema de Fermat y divida $f(x)$ por $x^p - x$]

16.

17. (25 puntos) Se considera el anillo $(\mathbb{Z}_{40}, +, \cdot)$ de los enteros módulo 40. Se representan las letras desde " A " a " Z ", el espacio en blanco " $\square$ " el punto " $\cdot$ " , el signo de interrogación " ? " y los dígitos de " 0 "a " 9 " por los números siguientes:
A$\mapsto$ 0, B$\mapsto$ 1, C$\mapsto$ 2, D$\mapsto$ 3, E$\mapsto$ 4, F$\mapsto$ 5, G$\mapsto$ 6, H$\mapsto$ 7, I$\mapsto$ 8, J$\mapsto$ 9,
K$\mapsto$ 10, L$\mapsto$ 11, M$\mapsto$ 12, N$\mapsto$ 13, $\tilde{\text{N}}\mapsto$ 14, O$\mapsto$ 15, P$\mapsto$ 16, Q$\mapsto$ 17, R$\mapsto$ 18,
S$\mapsto$ 19, T$\mapsto$ 20, U$\mapsto$ 21, V$\mapsto$ 22, W$\mapsto$ 23, X$\mapsto$ 24, Y$\mapsto$ 25, Z$\mapsto$ 26,
$\square\mapsto$ 27, $\cdot\mapsto$ 28, ?$\mapsto$ 29, 0$\mapsto$ 30, 1$\mapsto$ 31, 2$\mapsto$ 32, 3$\mapsto$ 33, 4$\mapsto$ 34,
$5\mapsto 35,\ 6\mapsto 36,\ 7\mapsto 37,\ 8\mapsto 38,\ 9\mapsto 39$

(a) Se encripta con un sistema afín que a cada letra le hace corresponder primero el número x que le corresponda en la tabla anterior, luego al número x se lo multiplica por a y se le suma b obteniendo un número $y = ax + b \, mod \, 40$. Finalmente a y se le asigna la letra o símbolo correspondiente en la tabla. La pareja de números (a, b) es la clave de encriptar.
EJEMPLO: si $a = 13$ y $b = 18$ a " C " le corresponde $x = 2$ que se transforma en $y = 13x + 18 = 26 + 18 = 44 = 4 \, mod \, 40$, finalmente a 4 le asignamos la letra " E ".
Si la clave es $(a, b) = (13, 18)$ encriptar el mensaje " HOLA ".

(b) Descifrar el mensaje cifrado con la clave $(a, b) = (13, 18)$ dado por

ÑRÑR

(c) ¿Qué condicion debe cumplir a para que al cifrar con la clave (a, b) *siempre* pueda descifrarse el mensaje en forma unívoca?

18. Se recuerda que $\varphi(n)$ es la función que da el cardinal del conjunto $\{x \in \mathbb{N} : 0 \leq x \leq n;\ MCD(x, n) = 1\}$ y vale $\varphi(n) = n(1-1/p_1)(1-1/p_2)\cdots(1-1/p_k)$ donde $p_1, \ldots p_k$ son los **distintos** factores primos de n. Ejemplo: si $n = 100 = 2^2 \times 5^2$ entonces $\varphi(100) = 100(1 - 1/2)(1 - 1/5) = 40$
a) Hallar todos los valores de $\varphi(n)$ para n entre 90 y 100.

b) Hacer una lista de todos los números n para los que $\varphi(n) \leq 12$ y mostrar que esa lista es completa.

19. Para beneficio del lector que no posea una computadora elegimos números pequeños. Elegimos como alfabeto las 26 letras usadas en inglés, desde la "A" a la "Z". Vamos a encriptar en RSA usando trigrafos (de a 3 letras), y enviarlo como bloques de 4 letras.
    a) Para enviar el mensaje "HOY" al usuario A con clave pública $(n_A, e_A) = (46927, 17)$ primero hallamos el equivalente numérico de "HOY" que da ($a \mapsto 0$, $b \mapsto 1, \ldots,$ etc.) $7 \cdot 26^2 + 14 \cdot 26 + 25 = 5121$. Aplicando el método de exponenciación rápida se tiene que calcular el mensaje encriptado $c = m^{e_A} \, mod \, n_A$ a enviar.
    Factorizar $n_A = 46927$ y hallar la clave secreta de A, d_A. Comprobar que $c^{d_A} \, mod \, n_A$ es m, el mensaje original.

20. La clave publica usada por el Banco de Pelotas en la localidad de Cerro Bajo es $N = 10403$ y $e = 8743$. Las computadoras del Banco recibieron de fuente indeterminada el siguiente mensaje:

$$4746 - 8214 - 9372 - 9009 - 4453 - 8198$$

    ¿Que dice el mensaje que recibió la sucursal del Banco de Pelotas?
    Nota: Se representan las letras desde " A " a " Z " y el espacio en blanco " $\square$ " así:
    A$\mapsto$ 10, B$\mapsto$ 11, C$\mapsto$ 12, D$\mapsto$ 13, ..., Y$\mapsto$ 34, Z$\mapsto$ 35, $\square \mapsto$ 99
    No se incluye la $\tilde{\text{N}}$.

21. Sabiendo que $N = 3552327 = pq$ y que $\varphi(N) = 3548580$, factorizar N.

22. Sea n un entero positivo y p un factor primo de n.

    (a) Pruebe que $p - 1$ divide a $\varphi(n)$

    (b) Muestre que puede ocurrir que p no divida a $\varphi(n)$

    (c) Muestre que $n^2 > (\varphi(n))^2 > n$ para todo $n > 2$

    (d) Hallar todos los valores de n para los cuales $\varphi(n) = 18$. Repetir para los casos $\varphi(n) = 10$ y $\varphi(n) = 14$.

23. (Sistema Vigenère de encriptar)

Por varios cientos de años el sistema mas popular de encriptar fue el llamado Vigenère y sus variantes. Este es una generalización del método Cesar. La idea del mismo es mirar bloques de k letras como vectores de $\mathbb{Z}_N^k$ y transformar el texto sumando a cada vector $x \in \mathbb{Z}_N^k$ un vector fijo $b \in \mathbb{Z}_N^k$ (la clave), $x \to x + b \; mod \; N$. Se parte entonces el texto en grupos de k caracteres, cada carácter se representa como un elemento de $\mathbb{Z}_N$ y la clave también. Ejemplo: Con los representación A$\mapsto$ 0, B$\mapsto$ 1, C$\mapsto$ 2, D$\mapsto$ 3, ..., Ñ$\mapsto$ 14, ..., Y$\mapsto$ 25, Z$\mapsto$ 26, $\square\mapsto$ 27, usando $k = 3$ y como clave la palabra "CID"tenemos que b es en $\mathbb{Z}_{28}^3$, $b = (2, 8, 3)$. Para cifrar "VOLVERAN LAS OSCURAS GOLONDRINAS"partimos el texto en 3-grafos

"VOL VER AN$\square$ LAS $\square$OS CUR AS$\square$ GOL OND RIN AS"

Al final, en "AS", falta un carácter. Optamos por aplicarle la transformación solo a las letras que alcancen. Otro método podría ser rellenar el texto con caracteres *al azar* hasta que quede múltiplo de 3 (en general de k). "VOL VER AN$\square$"se corresponde con los vectores

$$(22, 15, 11),\ (22, 4, 18),\ (0, 13, 27)$$

de $\mathbb{Z}_{28}^3$ que al sumarles $b = (2, 8, 3)$ en $\mathbb{Z}_{28}^3$ dan respectivamente:

$$(24, 23, 14),\ (24, 12, 21),\ (2, 21, 2)$$

. Llevado a texto queda:

"XWÑXMUCUC". Análogamente con el resto del mensaje. Este método de encriptar no resiste el ^{a}nálisis de frecuencias". Existen modos de "calcular k"y luego ver que caracteres son más frecuentes en la subsucesión de caracteres $a_i, a_{i+k}, a_{i+2k}, \ldots$ para $i = 1, 2, \ldots, k$. En Castellano los caracteres más frecuentes son "E"con una frecuencia de 13,15%, "A"con una frecuencia de 12,69%, "O"con una frecuencia de 9,49%, "S"con una frecuencia de 7,60%, "N"con una frecuencia de 6,95%, "R", "I", "L", y "D"con aproximadamente un 6% cada una. Esto sin contar que el espacio en blanco entre palabras, es más frecuente todavía.

Se sabe que el texto de más abajo está en Castellano y ha sido cifrado con el sistema Vigenère con $k = 3$ y la representación de las letras dada más arriba

(incluido el espacio en blanco). Descifrarlo.
”□SFKOOKPOXGVOKOKOZLQIBEÑCIÑFYSVZSKE
ÑFIFKOOKJWQYFLDRODRÑQÑAYWTSHODDLWOB”

jlvb