Números enteros, relojes y criptografía [*]

María José Felipe, Francisco Monserrat, Víctor Manuel Ortiz Sotomayor

"El joven estudiante universitario se ve confrontado, al comenzar sus estudios, con problemas que no guardan relación con las cosas que eran importantes en el instituto. Y, en consecuencia, se olvida pronto y por completo de todas ellas. Pero después de terminar sus estudios universitarios pasa a ser profesor y entonces se ve obligado de pronto a enseñar las tradicionales matemáticas elementales (...) *Y como sin ayuda no es capaz de encontrar un nexo que ponga en relación esta tarea con las matemáticas universitarias, pronto cae en una forma de enseñar avalada por el tiempo, y sus estudios universitarios pasan a ser un recuerdo más o menos agradable pero que no ejerce influencia en su enseñanza."*

Félix Klein[**]

Presentado en eXIDO18 (2018)

1. Introducción

Los rápidos avances científico-tecnológicos determinan un constante y raudo cambio tanto en la vida cotidiana como en la percepción del mundo que nos rodea, lo cual implica una necesidad de adaptación del conocimiento a dicho acelerado progreso. Sin embargo, los contenidos curriculares que se imparten en las asignaturas de matemáticas a lo largo de la Educación Secundaria (tanto en la ESO como en Bachillerato) adolecen, a nuestro entender, de un evidente estancamiento y no evolucionan de forma acorde a dicho progreso. Cuando a un alumno de Educación Secundaria se le menciona la palabra "matemática", en su mente aparecen, fundamentalmente, conceptos tales como ecuaciones, límites, derivadas, integrales, etc., todos ellos relacionados con la "matemática del continuo". Sin embargo, existen muchos aspectos de la matemática "no continua" que, bien por su belleza o por su utilidad, podrían tener un hueco más predominante en el currículum: teoría de números, combinatoria o teoría de grafos, por citar algunos ejemplos, que permiten acercar al estudiante de forma elemental al mundo de la informática y programación a través de aplicaciones computacionales sencillas aprovechando la variada disponibilidad de softwares de acceso libre en la web.

Con estas ideas como telón de fondo, en este trabajo nos centraremos en una rama de la matemática que es esencial en el desarrollo tecnológico en plena era digital y que, sin embargo, con pocas excepciones, no se considera en los contenidos impartidos en nuestros institutos de Educación Secundaria. Hablamos de la Matemática Discreta. El ojetivo de este artículo es mostrar un taller de matemáticas, dirigido principalmente a estudiantes de últimos cursos de secundaria, así como a estudiantes universitarios que se inician en dicha materia, en el que se desarrolla un aspecto de la matemática discreta muy relevante por sus aplicaciones a la tecnología moderna en el ámbito de la seguridad informática como es la "aritmética modular". Desde el punto de vista de la motivación, resulta altamente interesante que la introducción de estos nuevos conocimientos venga motivada por su aplicación en un ámbito

[*]Este trabajo ha sido financiado por el Proyecto de Innovación Docente PID-DMA 2018 "Modelizaciones algebraicas de algunos juegos matemáticos", del Departamento de Matemática Aplicada (Universitat Politècnica de València).

[**]Texto extraído de [3].

relevante. En nuestro caso, la aplicación vendrá dada por el método criptográfico RSA, tan vigente y utilizado hoy en día en cualquier proceso de firma digital y de autenticación de documentos. No es nuestra pretensión presentar una unidad didáctica completa y "rígida" sobre aritmética modular y criptografía, sino más bien presentar un "enfoque" de introducción de dichos conceptos, que el profesor puede adaptar y modificar de acuerdo con las metodologías docentes que considere oportunas y al tipo de alumnado al que va dirigido. Hemos de mencionar también la elección de este criptosistema por su facilidad de entendimiento para los estudiantes, aunque también debemos destacar la existencia de otros métodos alternativos e igualmente seguros que se utilizan en la actualidad y que también usan teoría modular en sus planteamientos, como es el caso de los métodos criptográficos basados en la teoría de *curvas elípticas* (véase Capítulo 6 del libro [4]).

Para realizar los cálculos computacionales, hemos hecho uso del sistema de libre acceso GAP (Groups, Algorithms and Programming) [1]. Se trata de un sistema de álgebra discreta computacional que proporciona un lenguaje de programación, una librería de funciones implementadas con algoritmos en dicho lenguaje, y amplias librerías de objetos algebraicos usados en enseñanza e investigación, tales como grupos y sus representaciones, anillos, espacios vectoriales, álgebras, estructuras combinatorias, autómatas y lenguajes, etc. De esta forma, tenemos una doble ventaja. Por un lado, la de utilizar la tecnología disponible a nuestro alcance, fomentando los conocimientos informáticos y el manejo de dispositivos digitales, imprescindibles en la actualidad para el futuro desarrollo personal y profesional de los estudiantes. Por otro lado, la de poder realizar operaciones que serían totalmente imposibles de hacer en un aula, bien por ser demasiado tediosas, o bien por no tener el alumno los conocimientos necesarios para poder realizarlas. Se usarán solo unos pocos comandos de GAP: aquellos estrictamente necesarios en los ejemplos y actividades que proponemos. Señalar que se hará uso de algunas funciones sencillas definidas ad-hoc, aunque solo es suficiente que el alumno entienda cuáles son sus inputs y outputs, sin entrar en detalle en lo que internamente hacen. Sería necesario, sin embargo, dedicar unos pocos minutos a proporcionar al alumno una introducción muy somera del programa GAP. Esta introducción queda fuera de las pretensiones de este artículo, pero puede encontrarse en internet información más que suficiente al respecto (ver [1]).

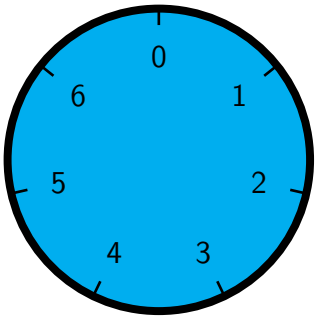
De forma general, proporcionaremos, a lo largo de las secciones siguientes, una explicación y secuenciación de los contenidos que se ajusten a las necesidades de un alumno de Bachillerato. En letra itálica sugeriremos, en momentos adecuados, la realización de ciertas actividades o ejercicios para que alumno afiance los conocimientos que se mencionan. El lector más interesado en Criptografía y Codificación de la Información, puede consultar [2, 4, 5]. La Sección 2 está dedicada a enfocar la introducción de conceptos muy básicos sobre aritmética modular. En realidad, los estrictamente necesarios para poder entender los contenidos de Criptografía que se proporcionarán como aplicación en secciones posteriores. Con el objetivo de focalizar la atención en lo realmente importante y útil en el desarrollo del método RSA, se ha tratado, deliberadamente, de omitir notaciones innecesarias que, aunque son de uso común por los matemáticos, no serán necesarias en la exposición que se da aquí. Presentaremos también un sencillo algoritmo con GAP que nos permitirá pasar un número en base 10 a una base cualquiera y que será necesaria para la resolución de las actividades propuestas. En la Sección 3 se explica el sistema criptográfico RSA. Obviamente, no se pretende proporcionar un desarrollo exhaustivo del mismo, sino únicamente proporcionar los datos estrictamente necesarios para que el alumno aprecie la utilidad de la aritmética modular. Así pues, se han omitido los enunciados y pruebas matemáticas que soportan el método, centrando la atención en el algoritmo en sí. En la Sección 4 incluimos un ejemplo explicativo de cifrado y descifrado de mensajes alfabéticos que ilustran los conceptos tratados con anterioridad. Finalmente, proponemos actividades complementarias para afianzar los contenidos mostrados sobre el cifrado RSA y para que puedan trabajar los alumnos los temas tratados en el taller.

2. Una breve incursión en la aritmética modular

2.1. Empaquetando números enteros usando relojes

2.1.1. Definición informal

Para cualquier número natural n, consideremos los elementos del conjunto $\{0, 1, 2, 3, \ldots, n-1\}$ dibujados como marcas (igualmente espaciadas) en una circunferencia. Denominemos n-reloj a una figura como esta. Por ejemplo, la figura siguiente muestra un 7-reloj: cada marca representa un valor entre 0 y 6.



Veremos cómo dividir el conjunto de los números enteros en **paquetes** usando un 7-reloj como este.

Escojamos un número entero cualquiera, por ejemplo el 23, y asociémoslo con una de las marcas del 7-reloj (marcas $0, 1, 2, 3, 4, 5$ ó 6) de la manera que explicamos a continuación. Imaginemos que este 7-reloj está dotado de una única varilla apuntando inicialmente al 0, y hacemos avanzar la varilla 23 posiciones en sentido horario (que consideraremos como "sentido positivo"). Si hacemos esto, la varilla da 3 vueltas completas al 7-reloj y avanza dos posiciones más, situándose finalmente en la marca 2. Asociaremos, por tanto, el número 23 con la marca 2. Procediendo de la misma manera podemos asociar, por ejemplo, el 7 con la marca 0, el 15 con la marca 1, el 27 con la marca 6, etc. Como seguramente se haya deducido, calcular la marca del 7-reloj asociada a cualquier número positivo es bien fácil: solo hay que dividir el número entre 7 y ver cuál es el resto.

Pero, ¿qué ocurre cuando el número entero es negativo? Por ejemplo, consideremos el número -17. En este caso procederemos de manera análoga a la explicada en el párrafo anterior, pero en lugar de mover la varilla en sentido horario (positivo), la moveremos en sentido antihorario (que, para nosotros, será el "sentido negativo"). Si hacemos esto, la varilla dará 2 vueltas completas al 7-reloj (avanzando 14 posiciones en sentido negativo) y avanzará 3 posiciones más, situándose sobre la marca 4. Asociaremos, pues, el número -17 con la marca 4. De la misma manera asociaremos el -8 con la marca 6, el -13 con la marca 1, el -15 con la marca 6, etc.

Procediendo de esta manera, asociamos cada número entero con cada una de las marcas del 7-reloj (denotadas por 0, 1, 2, 3, 4, 5 y 6). Dicho de otra manera, dividimos el conjunto de números enteros en 7 paquetes (cada uno de ellos correspondiente a una de las marcas).

El paquete correspondiente a la marca 0 está formado por todos los múltiplos de 7 (positivos y negativos, e incluyendo el 0). Esto es consecuencia de que la única forma de que la varilla se sitúe en la marca 0 es que se den varias vueltas completas al 7-reloj (en sentido positivo o negativo), pero sin más avances "extras". Esto solo es posible si el número entero es un múltiplo de 7.

El paquete correspondiente a la marca 1 está formado por todos los enteros que siguen el siguiente patrón: "múltiplo de $7 + 1$". Por ejemplo, el 1 (que es $0 \cdot 7 + 1$), el -6 (que es $(-1) \cdot 7 + 1$), el 15 (que es $2 \cdot 7 + 1$), etc.

El paquete correspondiente a la marca 2 está formado por los enteros de la forma "múltiplo de $7 + 2$". Por ejemplo, el 9 (que es $1 \cdot 7 + 2$), el 16 (que es $2 \cdot 7 + 2$), el -19 (que es $(-3) \cdot 7 + 2$), etc.

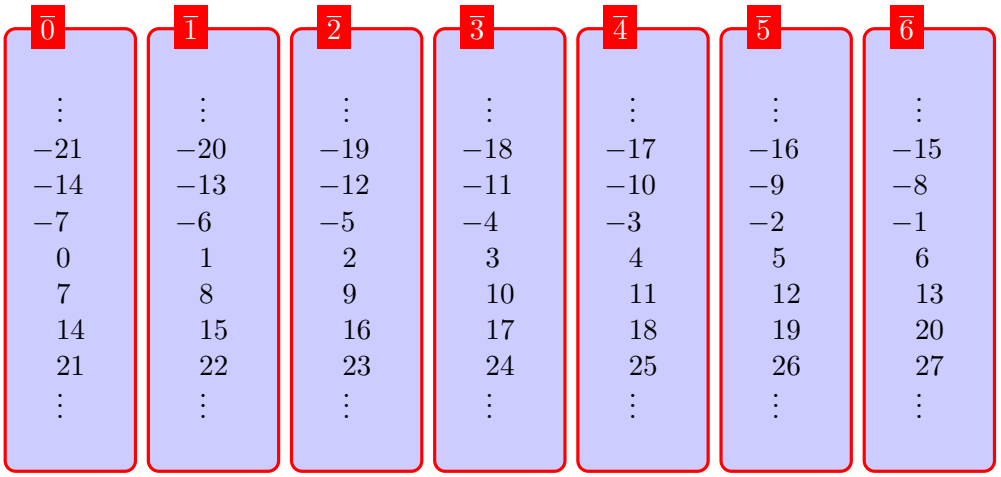
Procediendo sucesivamente de esta manera se describen los 7 paquetes en que hemos dividido el conjunto de los números de enteros.

2.1.2. Formalización

El número de marcas del reloj (7, en este caso) se denomina **módulo** y cada uno de los paquetes se denomina **clase**, o también **clase módulo 7** (si queremos especificar el módulo)[1]. Cada número entero pertenece a una (y solo una) clase, que se denota escribiendo una barra encima del número. Por ejemplo, la clase del 0 se denotará por $\overline{0}$ y diremos que es un **representante de su clase**, la clase del 23 se denotará por $\overline{23}$ y diremos que 23 es un representante de su clase, etc. Observa que, por ejemplo, las clases del 23 y del 2 son la misma (la posición de la varilla avanzando 2 posiciones en sentido positivo es la misma que avanzando 23). Luego 23 y 2 son representantes de la misma clase. Por tanto, podemos escribir $\overline{23} = \overline{2}$. Obsérvese que la diferencia $23 - 2$ es múltiplo de 7. Denotaremos por $\mathbb{Z}_7$ al conjunto de las clases módulo 7. Es decir:

$$\mathbb{Z}_7 = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}, \overline{6}\}.$$

En la figura siguiente hemos representado, de manera esquemática, cada una de estas clases.



De la misma manera en la que hemos definido las clases módulo 7, podemos definir las clases módulo n, para cualquier entero positivo n. El conjunto de estas clases será

$$\mathbb{Z}_n = \{\overline{0}, \overline{1}, \ldots, \overline{n-1}\}.$$

Observa que, dentro de cada clase, existe un único número entero entre 0 y $n-1$, que denominaremos **representante principal de la clase**.

✠ *Sugerimos, después de explicar estos primeros conceptos, proponer una serie de ejercicios sobre cálculo de representantes principales de clases, para diferentes valores del módulo.*

2.2. Sumando clases

En esta sección veremos cómo sumar clases. Continuaremos con el mismo ejemplo de la sección anterior (el conjunto de clases módulo 7), aunque todo puede reproducirse de manera análoga para cualquier módulo n.

¿Cómo podemos "sumar" las clases $\overline{5}$ y $\overline{6}$? La manera es bastante natural e intuitiva: simplemente situaremos la varilla del 7-reloj en la posición 0, avanzaremos 5 posiciones en sentido positivo, y 6

[1] El término matemático preciso es el de **clase de congruencia módulo 7**, pero lo omitiremos para simplificar la exposición.

posiciones más (también en sentido positivo). Si hacemos esto veremos que la varilla se sitúa en la marca $\overline{4}$, con lo cual diremos que:

$$\overline{5} + \overline{6} = \overline{4}.$$

Observa que avanzar la varilla 5 posiciones y luego 6 posiciones más equivale a avanzarla $5 + 6 = 11$ posiciones de una sola vez. Por tanto, otra manera de realizar la suma anterior es la siguiente: calculamos primero la clase de $5+6$ y calculamos luego su representante principal (que es 4). Es decir:

$$\overline{5} + \overline{6} = \overline{5+6} = \overline{11} = \overline{4}.$$

De esta manera podemos definir la suma de dos clases $\overline{a}$ y $\overline{b}$ cualesquiera de $\mathbb{Z}_7$:

$$\overline{a} + \overline{b} = \overline{a+b}.$$

Para que esta operación suma esté bien definida, no debería depender de los representantes elegidos de cada clase que sumamos. Por ejemplo, debería de valer lo mismo $\overline{5} + \overline{6}$ que $\overline{12} + \overline{-1}$, puesto que $\overline{5} = \overline{12}$ y $\overline{6} = \overline{-1}$. Puedes comprobar que ambos resultados coinciden. De hecho, puede demostrarse que esto ocurre siempre y, por tanto, la operación está bien definida.

Veamos ahora qué propiedades interesantes satisface la suma de clases. No haremos una prueba formal de las propiedades, pero es conveniente que te convenzas de ellas proponiéndote varios ejemplos. En primer lugar, se satisfacen la propiedades **asociativa** y **conmutativa**. Es decir, dadas tres clases cualesquiera $\overline{a}, \overline{b}, \overline{c}$ de $\mathbb{Z}_7$,

$$(\overline{a} + \overline{b}) + \overline{c} = \overline{a} + (\overline{b} + \overline{c}) \quad \text{y} \quad \overline{a} + \overline{b} = \overline{b} + \overline{a}.$$

La suma de clases posee **elemento neutro** (es decir, una clase que, al sumársela a cualquier otra, "no produce ningún efecto" sobre la misma). Es la clase del 0:

$$\overline{a} + \overline{0} = \overline{a}.$$

Finalmente, cualquier clase $\overline{a}$ posee **elemento simétrico** (llamado también **opuesto** por tratarse de la operación suma)[2], es decir, otra clase que, al sumársela a $\overline{a}$, se obtiene el elemento neutro $\overline{0}$. El opuesto de $\overline{a}$ es $\overline{-a}$ (la clase de $-a$):

$$\overline{a} + \overline{-a} = \overline{0}.$$

Por ejemplo, el opuesto de $\overline{2}$ en $\mathbb{Z}_7$ es $\overline{-2}$, que coincide con $\overline{5}$.

✠ *Sugerimos, en este punto, la realización de varios ejercicios para practicar la suma de clases, para distintos valores del módulo, y verificación de sus propiedades.*

2.3. Multiplicando clases

De manera similar a la suma, podemos definir el **producto** de clases. Dadas dos clases $\overline{a}$ y $\overline{b}$ definimos la clase producto, $\overline{a} \cdot \overline{b}$ como la clase del producto de a y b. Es decir:

$$\overline{a} \cdot \overline{b} = \overline{a \cdot b}.$$

Por ejemplo, siguiendo con $\mathbb{Z}_7$, el producto $\overline{2} \cdot \overline{5}$ es la clase de 10, que coincide con la clase de 3. Es decir:

$$\overline{2} \cdot \overline{5} = \overline{3}.$$

[2]En general, decimos que $(\mathbb{Z}_7, +)$ es un grupo abeliano.

Al igual que en el caso de la suma, puede probarse que el producto está bien definido, es decir, no depende de los representantes elegidos en cada clase. Veamos ahora qué propiedades satisface el producto de clases. Al igual que en el caso de la suma, únicamente enunciaremos las propiedades sin demostrarlas. En primer lugar, se satisfacen las propiedades asociativa y conmutativa. Es decir, dadas tres clases cualesquiera $\overline{a}, \overline{b}, \overline{c}$,

$$(\overline{a} \cdot \overline{b}) \cdot \overline{c} = \overline{a} \cdot (\overline{b} \cdot \overline{c}) \quad \text{y} \quad \overline{a} \cdot \overline{b} = \overline{b} \cdot \overline{a}.$$

El producto de clases también posee **elemento neutro**, que es la clase del 1, ya que:

$$\overline{a} \cdot \overline{1} = \overline{a}.$$

¿Y qué pasa con el elemento simétrico? ¿Cualquier clase tiene elemento simétrico respecto del producto? Es decir, dada una clase $\overline{a}$, ¿existe otra clase $\overline{b}$ tal que $\overline{a} \cdot \overline{b} = \overline{1}$? En caso de existir, esta clase $\overline{b}$ se denomina, al tratarse de la operación producto, **inverso** de $\overline{a}$, y se denotará por $\overline{a}^{-1}$.

Evidentemente, la clase de 0 no puede tener inverso, ya que $\overline{a} \cdot \overline{0} = \overline{0}$ para cualquier clase $\overline{a}$. La clase de 1 sí que tiene: su inverso es ella misma, puesto que $\overline{1} \cdot \overline{1} = \overline{1}$. Veamos qué pasa con las restantes clases en el caso de $\mathbb{Z}_7$. Para ello resultará útil construir la llamada *tabla de Cayley* del producto en $\mathbb{Z}_7$. Se trata de una tabla cuyas filas y columnas se corresponden con los elementos de $\mathbb{Z}_7$ y, en cada casilla, se representa el producto de las clases correspondientes a la fila y la columna asociadas a esa casilla. La tabla de Cayley respecto al producto $\mathbb{Z}_7$ es la siguiente[3]:

$\cdot$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Observando la tabla vemos que $\overline{2}$ tiene inverso, y su inverso es $\overline{4}$, pues $\overline{2} \cdot \overline{4} = \overline{1}$. De la misma manera observamos que $\overline{3}^{-1} = \overline{5}$, $\overline{4}^{-1} = \overline{2}$ and $\overline{5}^{-1} = \overline{3}$ y $\overline{6}^{-1} = \overline{6}$. Por tanto, en este caso, todos los elementos de $\mathbb{Z}_7$, excepto $\overline{0}$, tienen inverso. Sin embargo, **este hecho no es cierto en general**, como veremos a continuación. Consideremos ahora el conjunto $\mathbb{Z}_4 = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}\}$ y escribamos la tabla de Cayley correspondiente a su producto:

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Como puedes observar, **la clase de 2 no tiene inverso** en $\mathbb{Z}_4$. Nos planteamos la siguiente pregunta: ¿qué clases tienen inverso en $\mathbb{Z}_n$? La respuesta viene dada en la siguiente propiedad, que no demostraremos:

Una clase $\overline{a}$ en $\mathbb{Z}_n$ tiene inverso si, y solo si, a y n no tienen divisores primos comunes (es decir, si $\text{mcd}(a, n) = 1$).

[3]En general, decimos que $(\mathbb{Z}_7, +, \cdot)$ es un anillo conmutativo.

✠ *Se propondrá al alumno, en este punto, la realización de las tablas de Cayley del producto en* $\mathbb{Z}_2$, $\mathbb{Z}_5$, $\mathbb{Z}_6$ *y* $\mathbb{Z}_{12}$, *indicando qué elementos tienen inverso y los inversos de los mismos. Se requerirá que comprueben la propiedad anterior en estos ejemplos. Se planteará también la cuestión siguiente: ¿para qué valores de* n *se verifica que todo elemento (excepto la clase del cero) tiene inverso?*

2.4. Cambio de base de un número entero

Otro concepto del que vamos a hacer uso en el descifrado de mensajes encriptados es el cambio de base de un número entero. Los números enteros vienen dados en base 10, lo cual significa que, por ejemplo, el número $1369 = 1 \cdot 10^3 + 3 \cdot 10^2 + 6 \cdot 10 + 9$. Sin embargo, dicho número en base 26 viene dado por la terna $(2, 0, 17)$ ya que $1369 = 2 \cdot 26^2 + 0 \cdot 26 + 17$. El pasar de una base b a la decimal es pues sencillo, sin embargo para realizar la conversión contraria, de un número en base 10 a una base b, se requiere realizar divisiones sucesivas de dicho número por la base y tener en cuenta al final las cifras correspondientes a los diferentes restos que hemos ido obteniendo en dichas divisiones, como se puede ver por ejemplo en la siguiente figura[4]:

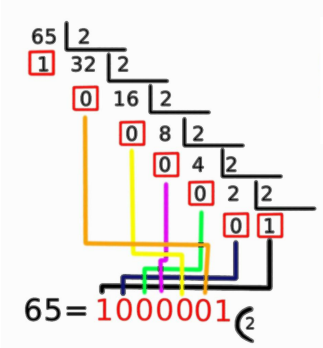


Figura 1: Conversión del numero 65 en base 10 a base 2 (binaria).

La siguiente función es un pequeño programa realizado en el lenguaje GAP que nos permite la conversión de un número entero x en base 10 a base b:

```
gap> Base:=function(x, b)
>       local r, q, l;
>       q:=QuoInt(x, b);
>       r:=RemInt(x, b);
>       l:=[r];
>       repeat
>               r:=RemInt(q, b);
>               q:=QuoInt(q, b);
>               Add(l, r);
>       until q<b;
>       Add(l, RemInt(q, b));
>       return Reversed(l);
> end;
function( x, b ) ... end
gap> Base(1369, 26);
[2,0,17]
```

[4]Imagen extraída de `https://goo.gl/images/tnRLzP`.

Sin entrar en demasiados detalles, hemos de comentar que la función anterior define tres variables locales mediante el comando `local`. Posteriormente calcula los respectivos cocientes y restos de dividir el número x (y sus cocientes sucesivos) entre b, mediante los comandos `QuoInt` ("quotient") y `RemInt` ("remainder"), hasta que algún cociente sea menor que el divisor b. Los sucesivos restos los va almacenando en el vector l, que en orden inverso nos proporciona los correspondientes coeficientes de x en base b. Así, por ejemplo, `Base(1369, 26);` nos da como resultado el vector `[2, 0, 17]`. Observemos además la igualdad de las clases $\overline{1369} = \overline{17}$ módulo 26.

✠ *Se sugiere realizar distintos ejercicios de cambio de base de números enteros y su comprobación con la función anteriormente definida en GAP.*

2.5. Exponenciación modular

Dentro del algoritmo criptográfico RSA, necesitaremos recurrentemente utilizar exponenciaciones modulares, i.e. calcular potencias de un número entero módulo otro número. A pesar de que los ejemplos que presentaremos en este trabajo considerarán solamente números "pequeños", hemos de tener en cuenta que en la realidad las operaciones involucran números de muchas cifras, por lo que dichas operaciones pueden resultar tediosas de realizar incluso para un ordenador. Por ejemplo, calcular con GAP el número 13789^{722341} en $\mathbb{Z}_{2345}$ que como máximo involucra enteros de 6 cifras nos lleva un tiempo de 2 minutos y 16 segundos, mientras que con el conocido método que presentamos a continuación el cálculo se realiza en milésimas de segundos.

Este método de eficiencia para calcular la exponenciación modular utiliza implícitamente la expresión del exponente en binario, ya que se trata de ir calculando potencias al cuadrado para ir reduciendo el número de multiplicaciones a realizar. Veámoslo mejor con el siguiente ejemplo: calcular n^{15} nos llevaría 14 multiplicaciones (de n con sí mismo), mientras que si realizamos $((n*n^2)^2*n)^2*n$ solamente calculamos 6 multiplicaciones. Implícitamente, hemos expresado 15 en binario que resulta $(1,1,1,1)$ (esto se puede calcular con la función definida en la subsección anterior), luego $15 = 2^3 + 2^2 + 2 + 1$ y posteriormente hemos calculado $n^{13} = n^{2^3} * n^{2^2} * n^2 * n = (n^{2^2})^2 * (n^2)^2 * n^2 * n = (n^{2^2} * n^2 * n)^2 * n = ((n^2 * n)^2 * n)^2 * n$. Este algoritmo se puede resumir de la siguiente manera:

$$n^a = \begin{cases} n, & \text{si } a = 1. \\ (n^{\frac{a}{2}})^2, & \text{si } a \text{ es par.} \\ n * n^{a-1}, & \text{si } a \text{ es impar.} \end{cases}$$

Programado en lenguaje de GAP de manera *recursiva* (esto es, el resultado final depende de otros resultados de la misma función de valores más "pequeños"), el algoritmo que calcularía n^a módulo m quedaría como sigue:

```
gap> Potencia:=function(n, a, m)
>       if a=1 then;
>               return n mod m;
>       elif RemInt(a, 2)=0 then;
>               return (Expo_binaria_mod(n, a/2, m)^2) mod m;
>       elif RemInt(a, 2)=1 then;
>               return (n*Expo_binaria_mod(n, a-1, m)) mod m;
>       fi;
> end;
function( n, a, m ) ... end
gap> Potencia(13789, 722341, 2345);
> HMSMSec(time);
2029
15
gap> (13789^722341) mod 2345;
> HMSMSec(time);
```

```
2029
[ 0, 2, 16, 236 ]
```

La función `Potencia(x, y, n)` que computa n^a en $\mathbb{Z}_m$ en GAP se ejecuta por casos: si el exponente es 1, entonces trivialmente el resultado es n módulo m (el comando `mod` calcula clases en módulo); si el exponente es par, entonces el resultado es $n^{\frac{a}{2}}$ módulo m (que lo calcularía mediante la misma función) y se eleva al cuadrado; finalmente, si el exponente es impar, entonces el resultado es multiplicar n por n^{a-1} módulo m y de nuevo el segundo término se calcularía por la misma función. Notemos que el algoritmo está bien definido, pues aunque se llame a sí mismo recursivamente, siempre hay un final que es precísamente cuando el exponente es 1.

Seguidamente, hemos calculado 13789^{722341} en $\mathbb{Z}_{2345}$ tanto directamente como por el método anterior descrito, y podemos comprobar que el resultado $\overline{2029}$ coincide en ambos casos, siendo el tiempo de cómputo (mediante el comando `HMSMSec(time)`) de 15 milisegundos mediante dicho algoritmo y de 2 minutos y 16 segundos mediante el cálculo directo.

2.6. La función de Euler

Como ya sabemos, si n es un número natural, un elemento de $\mathbb{Z}_n$ (es decir, una clase de congruencia $\overline{m}$ módulo n) es *invertible* si, y solo si, $\text{mcd}(m, n) = 1$. La función Phi de Euler, $\Phi : \mathbb{N} \to \mathbb{N}$, se define de la siguiente manera:

$$\begin{aligned} n \mapsto \Phi(n) &:= \text{Número de elementos invertibles de } \mathbb{Z}_n \\ &= \text{Número de enteros positivos menores que } n \text{ y coprimos con } n. \end{aligned}$$

Por ejemplo, $\Phi(6) = 2$ porque los enteros positivos coprimos con 6 y menores que 6 son 1 y 5. La función Φ es muy costosa de calcular en general. Sin embargo, posee algunas propiedades que pueden facilitar su cálculo en ciertas condiciones:

1. Si n es primo, entonces $\Phi(n) = n - 1$ y $\Phi(n^s) = n^{s-1}(n-1)$;
2. Si n y m son coprimos entre sí (es decir, si $\text{mcd}(n, m) = 1$), entonces $\Phi(nm) = \Phi(n)\Phi(m)$.

Estas dos propiedades permiten el cálculo de $\Phi(n)$ una vez se conozca la factorización de n en primos (de hecho, los problemas de calcular $\Phi(n)$ y de factorizar n se consideran "equivalentes"). Por ejemplo, si n es producto de dos primos que conocemos, $n = pq$, entonces $\Phi(n) = \Phi(pq) = \Phi(p)\Phi(q) = (p-1)(q-1)$, haciendo uso de las propiedades anteriores. Si los primos p y q son elegidos por nosotros, entonces el cálculo de $\Phi(n)$ es inmediato, pero si el valor de n es suficientemente grande y desconocemos los factores p y q, entonces el problema resulta muy complicado. Esta es la clave del uso de la función Φ en el sistema criptográfico RSA. El sistema GAP dispone de un comando que calcula la función Phi de Euler de cualquier número natural (que no sea excesivamente grande): `Phi`. Por ejemplo:

```
gap> Phi(20);
8
```

nos dice que $\Phi(20) = 8$. Si queremos ahora ver cuáles son dichos números coprimos con 20, podemos hacer uso del comando `PrimeResidues(n);`. Por ejemplo, para $n = 20$:

```
gap> PrimeResidues(20);
[ 1, 3, 7, 9, 11, 13, 17, 19 ]
```

✠ *Se propone el realizar diferentes ejercicios de cálculo de la función Φ para distintos números enteros, el cálculo de los números contabilizados por esta función y su comprobación con el sistema GAP.*

3. El método criptográfico RSA

El sistema criptográfico RSA debe su nombre a sus tres inventores, Ronald Rivest, Adi Shamir y Leonard Adleman, que publicaron por primera vez el método en 1977. Ha estado bajo patente de los Laboratorios RSA hasta el 20 de septiembre de 2000, por lo que su uso comercial estuvo restringido hasta esa fecha. Este sistema permite intercambiar información de forma segura entre varios participantes. Pertenece a la categoría de los sistemas de clave pública, lo que significa que puede ser usado por grupos bastante numerosos. Cada participante tiene dos claves: una clave privada, que utiliza para descifrar los mensajes que recibe, y una clave pública, que utilizan los demás participantes cuando desean cifrar un mensaje dirigido a él. Estas dos claves son inversas entre sí, pero debe ser computacionalmente imposible deducir la clave privada a partir de la pública (y en esto radica la seguridad del sistema). Para conseguir este objetivo se utilizan lo que se llaman *funciones de una vía*, esto es, procesos matemáticos que son bastante rápidos de realizar en un sentido, pero muy costosos de invertir. En el caso del método RSA, el problema matemático que dificulta esta inversión es la factorización de un número entero como producto de primos.

Como se sabe, todo número entero puede escribirse de forma única como producto de primos (Teorema Fundamental de la Aritmética). Esta escritura es muy fácil de obtener cuando el número a factorizar es pequeño. Por ejemplo, si queremos factorizar el número 229369 con GAP escribiremos lo siguiente:

```
gap> Factors(229369);
[ 7, 7, 31, 151 ]
```

obteniendo que la descomposición de 229369 en producto de factores primos es $7^2 \cdot 31 \cdot 151$. Ahora bien, cuando el número es grande las cosas cambian muchísimo: no hay forma conocida de obtener esta factorización en tiempos razonables. Es un problema que, supuestamente, es computacionalmente imposible.

El sistema RSA ha permanecido invulnerable hasta hoy, a pesar de los numerosos ataques de criptoanalistas. En él se basan la gran mayoría de las transacciones bancarias seguras por Internet, o la firma digital de documentos. Explicaremos a continuación el funcionamiento del sistema RSA, ilustrando la explicación con un ejemplo.

1. Cada usuario del sistema debe elegir una pareja de números primos p, q. Estos deben ser *suficientemente grandes*, concepto variable según la capacidad de computación del momento. Actualmente, el número primo más grande conocido es el quincuagésimo primo de Mersenne $2^{77,232,917} - 1$, con un total de 23,249,425 cifras, que fue descubierto el pasado 26 de diciembre de 2017 por el ingeniero eléctrico Jonathan Pace. El software GAP contiene una pequeña lista de 168 números primos correspondientes a los números primos menores que 1000 denotada por `Primes`. Así, el número más grande de la lista es el 997.

   ```
   gap> Primes[168];
   997
   ```

   Un ejemplo **nada realista**, dado lo irrisorio del tamaño de los datos, pero muy adecuado para comprender el funcionamiento del sistema, podría ser el que mostraremos a continuación: un usuario llamado Pepe considera los números primos siguientes (esta es una información "secreta" que **no** comparte con el resto de usuarios):

   $$p = 113, \qquad q = 229.$$

   Introducimos estos números primos en GAP, usando el comando "`:=`" para definir variables:

```
gap> p:=113;
113
gap> q:=229;
229
```

2. Se calculan los valores $n := pq$ y $r := \Phi(n) = (p-1)(q-1)$. Como ya comentamos anteriormente, la dificultad del sistema RSA radica en la dificultad de hallar $\Phi(n)$ cuando n es grande y desconocemos p y q. En nuestro ejemplo:

$$n = 113 \cdot 229 = 25877, \qquad r = 112 \cdot 228 = 25536.$$

Con GAP:

```
gap> n:=p*q;
25877
gap> r:=Phi(n);
25536
```

3. Se elige (arbitrariamente) un número entero e tal que $0 < e < r$ y $\mathrm{mcd}(e, r) = 1$. En nuestro ejemplo elegimos

$$e = 1321.$$

Puede comprobarse que e satisface las condiciones exigidas usando el comando `Gcd` de GAP, el cual calcula el máximo común divisor de dos números (Greatest Common Divisor):

```
gap> e:=1321;
1321
gap> Gcd(e,r);
1
```

4. La condición $\mathrm{mcd}(e, r) = 1$ significa que la clase de e, dentro del conjunto de las clases módulo r, $\mathbb{Z}_r$, tiene inverso. Calculamos d un representante principal de $\overline{e}^{-1}$. Con GAP puede hacerse de la siguiente manera: el comando `ZmodnZObj(a,b)` calcula la clase de a módulo b, posteriomente calculamos su clase inversa mediante el comando `Inverse()`, y finalmente `Int()` calcula un representante de dicha clase.

```
gap> d:=Int(Inverse(ZmodnZObj(e, r)));
17881
```

Por tanto, $\overline{e}^{-1} = \overline{17881}$ en $\mathbb{Z}_r$.

5. La *clave pública* del sistema RSA es el par (n, e). Esto significa que **este par de valores es conocido por todos los usuarios** y es el que va a permitir, a cualquier usuario, *cifrar* el mensaje que desea que solo sea descifrado por Pepe.

   La *clave privada* del sistema es el valor d. Esto significa que **el valor** d (calculado a partir de p y q) solo es conocido por Pepe (**no es compartido con el resto de usuarios**). Esta será la clave que permitirá a Pepe descifrar los mensajes cifrados que le envíen el resto de usuarios.

6. Los mensajes, tanto en claro como de descifrado, deben previamente identificarse con elementos del conjunto $\mathbb{Z}_n$ de clases módulo n. Veremos más adelante con un ejemplo formas de realizar dicha identificación.

   En nuestro caso, supongamos que el mensaje que una usuaria, llamada Juana, desea enviar a Pepe se identifica con la clase de $m = 12423$ en $\mathbb{Z}_n = \mathbb{Z}_{25877}$. Veamos cómo Juana debe cifrar el mensaje y cómo Pepe ha de descifrarlo.

7. **Proceso de encriptación:** para *cifrar* el mensaje m que quiere enviar Juana a Pepe se calcula el representante principal de la clase de m^e módulo n. El resultado de efectuar esta exponenciación modular será el mensaje encriptado. Podemos utilizar aquí el algoritmo explicado en secciones anteriores junto con su programación en GAP mediante la función **Potencia**. Aplicando esta función a nuestro ejemplo obtenemos:

```
gap> Potencia(12423,e,n);
14946
```

   Por tanto, la clase de 14946 módulo n es el mensaje *encriptado* que transmite Juana a Pepe.

   En resumen, para encriptar un mensaje hemos calculado la imagen de m por la función $f : \mathbb{Z}_n \to \mathbb{Z}_n$ definida por $f(x) = x^e$. Como la clave (n, e) es pública, todo el mundo, como Juana, podría encriptar fácilmente un mensaje para enviárselo a Pepe simplemente calculando la imagen por f. Sin embargo, *siempre que los primos tomados p y q sean suficientemente grandes* (cosa que no ocurre en este ejemplo) f es una *función de una vía*, es decir:

   - para todo elemento $x \in \mathbb{Z}_n$ es computacionalmente sencillo calcular $f(x)$,
   - dado $\overline{y} \in \mathbb{Z}_n$, es computacionalmente imposible, en general, determinar un elemento $\overline{x} \in \mathbb{Z}_n$ tal que $f(\overline{x}) = \overline{y}$.

   Es más, f es una *función trampa*, es decir, además de las propiedades anteriores tiene la propiedad adicional de que es biyectiva y su función inversa que se utiliza en *el descifrado* del mensaje (la *trampa*) permite calcular eficientemente el inverso por f de cualquier elemento de $\mathbb{Z}_n$.

8. **Proceso de descifrado:** El proceso de descifrar el mensaje viene determinado por el siguiente hecho: la función f anterior es biyectiva y su inversa $f^{-1} : \mathbb{Z}_n \to \mathbb{Z}_n$ viene definida por la fórmula:

$$f^{-1}(\overline{y}) = \overline{y}^d.$$

   Conocer f^{-1} equivale a conocer la clave privada d, **que solo posee Pepe** para descifrar el mensaje. La trampa radica en que la clave secreta d no puede conocerse a partir de (n, e) sin el conocimiento de $r = \Phi(n) = (p-1)(p-1)$ (recuérdese que d es el representante principal del inverso de la clase de e módulo $(p-1)(q-1)$), y conocer r equivale a saber factorizar n como producto de los primos p y q. Si estos primos se han elegido suficientemente grandes, n tendrá un valor tan grande que será (o, al menos, así se cree) computacionalmente imposible factorizarlo. Aquí radica la seguridad del criptosistema RSA.

   Finalmente, si Pepe quisiera descifrar el mensaje encriptado $\overline{y}$ enviado por Juana, no tendría más que calcular el representante principal de $f^{-1}(\overline{y})$. En nuestro caso:

$$f^{-1}(\overline{14946}) = \overline{14946}^d = \overline{14946}^{17881} = \overline{12423}.$$

   Este cálculo lo podemos efectuar, como antes, usando la función **Potencia** que hemos definido en GAP:

```
gap> Potencia(14946,d,n);
12423
```

   Observa que se obtiene el mensaje del valor de m original.

✠ *Se propone realizar diferentes ejercicios de suma, producto, potencias e inversos de clases de $\mathbb{Z}_n$, comprobando los resultados con GAP.*

4. Cifrado de mensajes alfabéticos

En el siguiente ejemplo veremos una manera de transformar un mensaje alfabético en claro (es decir, sin encriptar) en uno numérico, a fin de aplicar el método RSA para encriptarlo. Prescindiremos de signos de puntuación y elaboraremos los mensajes con las letras del alfabeto de 26 letras A,. . . ,Z (la $\tilde{\text{N}}$ la podemos sustituir, por ejemplo, por NY), que identificamos con el conjunto de clases $\mathbb{Z}_{26} = \{\overline{0}, \overline{1}, .., \overline{25}\}$. Así pues, A se identificará con $\overline{0}$, B con $\overline{1}$, etc. Los espacios en blanco convendremos en sustituirlos previamente con una X.

En primer lugar particionaremos el mensaje inicial en mensajes más pequeños de longitud constante k (es decir, de k letras), con el fin encriptar separadamente cada mensaje de longitud k. El valor de k será convenido entre todos los usuarios. Aunque, en aplicaciones reales, es conveniente que el valor de k sea grande, consideraremos en este ejemplo $k = 3$.

Supongamos que Juana quiere enviar el siguiente mensaje encriptado a Pepe:

$$URGENTE\ \ NECESITO\ \ DINERO$$

empleando la misma clave pública que en el ejemplo anterior:

$$(n, e) = (25877, 1321).$$

En primer lugar, como el espacio en blanco no figura entre los símbolos de nuestro alfabeto, lo sustituiremos por X:

$$URGENTEXNECESITOXDINERO$$

En segundo lugar añadiremos una X más al final del mensaje para que su longitud sea múltiplo de $k = 3$:

$$URGENTEXNECESITOXDINEROX$$

Así pues, el mensaje queda subdividido en 4 de longitud 3, que debemos encriptar separadamente:

$$URG\quad ENT\quad EXN\quad ECE\quad SIT\quad OXD\quad INE\quad ROX$$

Sustituiremos ahora las letras por números entre 0 y 25 teniendo en cuenta la identificación antes mencionada:

$$\begin{array}{cccc} 20\ 17\ 6 & 4\ 13\ 19 & 4\ 23\ 13 & 4\ 2\ 4 \\ 18\ 8\ 19 & 14\ 23\ 3 & 8\ 13\ 4 & 17\ 14\ 23 \end{array}$$

A continuación interpretaremos cada sub-mensaje (vector numérico) de longitud $k = 3$ mediante el número que tiene a ese mensaje por escritura en base 26. Así pues, el primer bloque 20 17 6 lo identificamos con la clase de:

$$20 \cdot 26^2 + 17 \cdot 26 + 6 = 13968$$

Obsérvese que este número determina totalmente el sub-mensaje 20 17 6: si quisiéramos recuperarlo, solo tendríamos que expresar en base 26 el número decimal 13968 y considerar la secuencia de coeficientes de las potencias de 26.

Haciendo lo mismo con los 3 sub-mensajes restantes se obtiene que el mensaje total puede ser identificado con el siguiente vector

$$(\overline{13968}, \overline{3061}, \overline{3315}, \overline{2760}, \overline{12395}, \overline{10065}, \overline{5750}, \overline{11879}).$$

Cada componente de ese vector se interpreta como una clase en $\mathbb{Z}_n = \mathbb{Z}_{25877}$ que puede encriptarse elevándola a $e = 1321$ y tomando el representante principal de la clase módulo n resultante. Con GAP

podemos hacer realizar este proceso utilizando la función `Potencia`. Se obtiene el siguiente vector, que constituye el mensaje cifrado que recibiría Pepe:

$$(\overline{13968}^e, \overline{3061}^e, \overline{3315}^e, \overline{2760}^e, \overline{12395}^e, \overline{10065}^e, \overline{5750}^e, \overline{11879}^e) =$$

$$= (\overline{13968}, \overline{23014}, \overline{717}, \overline{7749}, \overline{5798}, \overline{1806}, \overline{16247}, \overline{18476}).$$

¿Qué debería hacer Pepe para descifrarlo? Tomar su clave privada $d = 17881$, elevar cada componente del vector a d y tomar su clase módulo n.

$$(\overline{13968}^d, \overline{23014}^d, \overline{717^d}, \overline{7749}^d, \overline{5798}^d, \overline{1806}^d, \overline{16247}^d, \overline{18476}^d) =$$

$$= (\overline{13968}, \overline{3061}, \overline{3315}, \overline{2760}, \overline{12395}, \overline{10065}, \overline{5750}, \overline{11879})$$

Para transcribirlo a código alfabético, debería expresar cada componente del vector en base 26 e intercambiar cada cifra por la letra correspondiente. Por ejemplo, para la primera componente:

$$13968 = 20 \cdot 26^2 + 17 \cdot 26 + 6 \quad \Longrightarrow \quad 20 \;\; 17 \;\; 6 \quad \Longrightarrow \quad URG$$

La expresión, por ejemplo, de los números 13968 ó 3061 en base 26 puede calcularse con la función `Base` que hemos definido con GAP.

```
gap> Base(13968,26);
[20,17,6]
gap> Base(3061,26);
[4,13,19]
```

Se procedería análogamente con las restantes componentes.

Finalmente, Pepe decide compartir su clave privada con Juana y le envía el siguiente mensaje encriptado en contestación al mensaje recibido:

$$(\overline{7508}, \overline{15423}, \overline{16602}, \overline{4882})$$

¿Cuál ha sido la respuesta de Pepe?

Una observación importante que cabe señalar es que es necesario, al elegir n y k, que se satisfaga la desigualdad $26^k \leq n$, pues es lo que permite identificar cada sub-mensaje (número en base 26 de longitud k) con un elemento de $\mathbb{Z}_n$. En el caso del ejemplo: $26^k = 26^3 = 17576 < n = 25877$.

✠ *Proponemos para finalizar este taller la siguiente actividad como aplicación de lo comentado en las secciones anteriores para que el alumno afiance los contenidos adquiridos:*

Partiendo de dos números primos, podrían ser por ejemplo $p = 173$ y $q = 281$, calcula una posible clave pública (n, e) para el algoritmo RSA, así como su clave privada correspondiente. Comparte dichas claves con algunos compañeros en clase y cifra un mensaje; por ejemplo, podría ser el siguiente

I LOVE HOMER SIMPSON

Una vez descifrado dicho mensaje por tus compañeros, pueden contestarte mandándote un mensaje encriptado en contestación al que has enviado y que podrás descifrarlo posteriormente de acuerdo con las claves compartidas.

Indicaciones:

- Recuerda que debes sustituir el espacio por una "X" (o simplemente eliminarlo) para poder usar solo el alfabeto de 26 letras.
- Divide el mensaje en sub-mensajes de longitud $k = 3$ (añadiendo al final del texto las "X" necesarias para que la longitud sea múltiplo de $k = 3$) y encripta por separado cada sub-mensaje.
- Se verifica que $26^3 \leq n$, condición necesaria según la observación anteriormente mencionada.

Referencias

[1] The GAP Group. GAP - *Groups, Algorithms, and Programming.* Ver. 4.8.10. URL: `http://www.gap-system.org/`.

[2] L. Hernández Encinas, *La criptografía*, CSIC (2016).

[3] F. Klein, *Elementarmathematik vom höheren Standpunkt aus*, B.G. Teuber (3 volúmenes) (1908, 1909, 1928). Traducción al español: *Matemática elemental desde un punto de vista superior*, Nivola (2006).

[4] N. Koblitz, *A course in number theory and criptography*, Springer-Verlag (1987).

[5] C. Munuera, J. Tena, *Codificación de la información*, Universidad de Valladolid (1997).